Recomendações para melhoria da manutenabilidade de Sistemas Baseados em COTS

Aurea Matsumura Miyazawa

Trabalho Final de Mestrado

Instituto de Computação
Universidade Estadual de Campinas

Recomendações para melhorar a manutenabilidade de Sistemas Baseados em COTS

Aurea Matsumura Miyazawa

Novembro 2003

Banca Examinadora:

- Profa. Dra. Eliane Martins (orientadora)
- Profa. Dra. Cecília Mary Fischer Rubira
  Instituto de Computação - UNICAMP
- Profa. Dra. Rosângela A. D. Penteado
  Departamento de Computação - UFSCar
- Profa. Dra. Maria Beatriz Selgar de Toledo (Suplente)
  Instituto de Computação - UNICAMP

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Miyazawa, Aurea Matsumura

M699r Recomendações para melhoria da manutenabilidade de Sistemas Baseados em COTS / Aurea Matsumura Miyazawa -- Campinas, [S.P. :s.n.], 2003.

Orientador : Eliane Martins

Trabalho final (mestrado profissional) - Universidade Estadual de Campinas, Instituto de Computação.

1. Software – Manutenção. 2. Software – Qualidade. 3. Engenharia de software. I. Martins, Eliane. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Recomendações para melhorar a manutenabilidade de Sistemas de Software baseados em COTS

Este exemplar corresponde a redação final do Trabalho de Conclusao do curso de Mestrado Profissional em Computação devidamente corrigida e defendida por Aurea Matsumura Miyazawa e aprovada pela Banca Examinadora.

Campinas, 15 de Novembro de 2003.

Eliane Martins

Profa. Dra. Eliane Martins (orientadora)

Trabalho Final apresentado ao Instituto de Computação, UNICAMP, como requisito para a obtenção do título de Mestre em Engenharia de Computação.

TERMO DE APROVAÇÃO

Trabalho Final Escrito defendido e aprovado em 15 de dezembro de 2003, pela Banca Examinadora composta pelos Professores Doutores:

Profª. Drª. Rosângela Ap. Dellosso Penteado
UFSCar

Prof. Dr. Cecília M. Fischer Rubira
IC - UNICAMP

Profª. Drª. Eliane Martins
IC - UNICAMP

Abstract

A Commercial-Off-The-Shelf (COTS) software is an already-built product where functionalities can be acquired immediately, obtained at significantly lower price and developed by experts. From this point-of-view construct systems that integrate COTS products seems to be a great solution for a challenge: create software systems with a large number of functionalities and high technology in a reduced time-to-market.

But on the other hand, the handling of COTS products brings new challenges to the maintenance personnel who are required to evolve and enhance these systems. As software maintenance is long known as one of the most expensive and resource-requiring phase of the software development, we look for some strategies for building maintainable COTS systems.

First of all, we identified some risks associated with maintenance of COTS systems, and suggest some strategies that can be developed to promote the system maintainability. But the definitions for maintainability are many and its various nuances are often confused or misunderstood, therefore, we look for a maintainability concept that defines the characteristic of the maintainability requirement in component-based software systems in order to clarify the impact of maintainability on software systems.

We present in this study an association between each identified strategies, the minimized risks and the maintainability characteristics affected. Six abstraction levels, indicating when they must be implemented, grouped these strategies: Requirement, Project Planning, Coding, Testing and Maintenance, Configuration Management and Quality Control. This association attempts to assist project team to become aware of involved risks and to define when, where and how they should pay attention to the many faces of maintainability, contributing for a low-cost COTS system derived of a adequate maintenance phase.

Resumo

Produtos COTS (Commercial-Off-The-Shelf) são soluções prontas e disponíveis no mercado que reduzem o tempo de desenvolvimento e o custo do sistema, portanto integrá-los em um único sistema parece ser então, a grande solução para construir sistemas com várias funcionalidades e com tecnologia de ponta em tempo reduzido.

Mas por outro lado, o conceito destes sistemas integrados traz consigo diferentes desafios e riscos à equipe de manutenção, responsáveis pela alteração e correção de erros. Cientes da importância da fase de manutenção, procuramos neste estudo atividades que possibilitassem viabilizar a manutenção deste tipo de sistemas.

Analisando a literatura, levantamos os riscos associados à manutenção destes sistemas e as estratégias existentes para minimizá-los, porém as várias nuances sobre o termo "manutenabilidade" dificultavam entender objetivamente o impacto destas estratégias, o que nos levou a buscar a definição de "manutenabilidade" em termos de características da qualidade, tornando-a mais precisa.

Apresentamos neste estudo uma associação entre cada estratégia levantada, os riscos minimizados e as características da manutenabilidade afetadas. Estas estratégias foram agrupadas em 6 níveis de abstração, indicando a fase do desenvolvimento de um sistema que elas deverão ser realizadas.:Requisitos, Planejamento, Arquitetura, Testes e Manutenção, Gerência de Configuração e Controle de Qualidade. O objetivo desta associação é permitir que a equipe do projeto possa previnir-se dos riscos envolvidos, decidindo quando e como melhorar as várias faces da manutenabilidade, contribuindo desse modo para um sistema COTS de baixo custo derivado de uma fase de manutenção adequada.

Agradecimentos

A minha orientadora, Eliane, pela paciência, pela orientação e tempo dedicado.

Ao meu querido marido Flávio por toda compreensão, amor e apoio que tanto precisei para iniciar e finalizar essa tese. Em todos os momentos que pensei em desistir, lá estava ele me confortando e me incentivando a seguir em frente.

Aos meus dois amados filhos, André e Elisa que iluminam meu espírito com a felicidade de seus sorrisos, olhares e travessuras e me ensinam a ser uma pessoa melhor a cada dia.

Aos meus pais, Helena e Luiz, a minha irmã Luci e ao meu irmão Fábio, que apesar da distância que nos separam, com amor e carinho sempre me incentivaram a continuar meus estudos.

A minha grande e querida amiga Luciana pelo ombro, ouvido, incentivo, carona e companhia durante mais essa jornada.

E a Deus que me deu uma linda família com quem divido a alegria de ter terminado mais uma etapa em minha vida.

Sumário

Capítulo 1

Introdução

Atualmente usuários de sistemas estão cada vez mais exigentes em relação a um sistema de software, exigindo que ele tenha características como alto desempenho, tecnologia de ponta, multi-plataforma, além de várias funcionalidades. Para acompanhar estas tendências, desenvolvedores se vêem diante de uma gama de padrões, plataformas, linguagens e novas tecnologias que surgem a todo instante, mas que requerem treinamentos específicos além de tempo para desenvolvimento do sistema, o que pode inviabilizar algumas exigências do cliente: a rapidez na entrega do produto e o baixo custo.

A inviabilidade de construir sistemas desse porte fica mais clara ainda quando imaginamos que equipes especializadas deveriam ficar a disposição para correção de erros, alteração ou adição de funcionalidades, lembrando que seriam poucos clientes que se beneficiariam com essa solução.

Os produtos COTS (Commercial off-the-shelf) são sistemas pré-construidos nos quais o código-fonte não fica disponível, mas que podemos encontrar a venda em lojas especializadas como por exemplo: compiladores, gerenciadores de banco de dados, software criptográficos, browsers para internet.

Agregar vários destes produtos COTS em um único sistema parece ser então a grande solução deste problema, pois desse modo as empresas poderiam simplesmente selecionar os produtos COTS que estão disponíveis no mercado, adaptá-los e vendê-los, sem o inconveniente de precisar de mão-de-obra especializada. Some-se a isso, uma série de benefícios em relação ao seu uso, como por exemplo: permite que a equipe de desenvolvimento concentre-se apenas nos requisitos de alto nível do sistema, e tenham disponíveis produtos desenvolvidos conforme padrões modernos e atualizados disponíveis no mercado.

Mas apesar de todos estes benefícios, o uso de produtos COTS deve ser visto com cautela, pois exige a conhecimento de conceitos e riscos diferentes daqueles conhecidos no desenvolvimento tradicional, e casos eles não sejam tratados a economia gerada pela compra do produto COTS muitas vezes pode ser superada pelo gasto em integrar e manter o sistema coeso.

1.1.Motivação

Sommervile em [SOM03] afirma que a fase de manutenção geralmente é a que requer mais esforço e portanto a de custo mais elevado, e além disso a satisfação do cliente é diretamente proporcional a qualidade de manutenção oferecida pela organização. Além disso, Voas em [VOA00] citou que em 1997, 25,5% de um portifólio de uma típica corporação era composto de software COTS. Em 1998, esse número subiu para 28,9% e em 2002 estimou-se em 40%, constatando o crescente aumento da sua utilização.

Cientes da importância da fase de manutenção e da pouca literatura existente sobre sistemas que agregam produtos COTS, fomos motivados a estudar melhor os problemas que atingem a fase de manutenção destes sistemas, cientes de que apresentando soluções para esses problemas estaremos diminuindo o esforço e facilitando as atividades de manutenção, tendo portanto um impacto direto sobre o custo total do sistema, beneficiando tanto a organização que fez o sistema, quanto o usuário que recebe um produto com menor custo e maior qualidade.

1.2.Objetivos do Trabalho

Para sugerir atividades ou estratégias para viabilizar a manutenção em um sistema COTS, primeiramente identificamos quais os riscos associados a manutenção destes sistemas, procurando dessa forma conscientizar as empresas da existência deles. Uma vez identificados os riscos e as estratégias que aumentam a manutenabilidade de um sistema, surgiu um problema: as várias nuances sobre o termo "manutenabilidade" dificultavam entender objetivamente o impacto destas estratégias sobre a fase de manutenção. Essa dificuldade nos levou a buscar a definição de "manutenabilidade" [MAR03] em termos de características da qualidade e que clarificou esse impacto, permitindo que possamos compreender como cada estratégia poderia aumentar a manutenabilidade do sistema COTS.

Porém Vigder em [VIG97] ressalta que a manutenabilidade de um sistema não é uma característica que possa ser adicionada após o sistema ter sido construído, mas sim uma tarefa que deve ser considerada desde os primeiros estágios do desenvolvimento do sistema. Esta também é uma opinião compartilhada por Clapp e Taub em [CLA98], que também sugerem em seus trabalhos várias estratégias a serem feitas durante o desenvolvimento do sistema visando facilitar a manutenção do sistema.

Assim sendo, agrupamos as estratégias levantadas em seis níveis de abstração: Requisitos, Planejamento, Arquitetura, Testes e Manutenção, Gerência de Configuração e Controle de Qualidade. Estes níveis estão associados a fase do projeto em que a estratégia deve ser feita, permitindo que a equipe do projeto possa definir onde, como e quando se preocupar com as várias faces da manutenabilidade.

Apresentamos no final desta monografia uma tabela que associa para cada estratégia identificada, o risco minimizado por ela e o seu impacto na manutenabilidade do sistema. Ela pode ser utilizada como um guia, tanto para empresas que pretendam aumentar a manutenabilidade de seus sistemas realizando várias estratégias, quanto para empresas que procuram soluções para riscos específicos e, neste caso, executarão apenas uma ou duas estratégias. Além disso, a equipe do projeto terá controle sobre quais etapas do desenvolvimento do sistema deverão ser alteradas e defina a manutenção adequada para cada projeto, tendo em vista o custo benefício de cada estratégia apresentada.

Ressaltamos que este estudo aplica-se somente a sistemas que integram vários produtos em um único sistema, e não a sistemas constituídos de um único COTS e nem ao desenvolvimento de produtos COTS propriamente ditos. Além disso, procuramos não nos prendermos a nenhum modelo específico de desenvolvimento aumentando a aplicabilidade do estudo.

1.3. Organização do Texto

A estrutura do documento está dividida do seguinte modo:

O capítulo 2 descreve conceitos básicos sobre Manutenção de Sistemas, COTS, Sistemas COTS, manutenabilidade e padrões de projeto. O capítulo 3 descreve detalhadamente alguns dos principais riscos que surgiram com a utilização de produtos COTS. O capítulo 4 apresenta algumas abordagens atuais que discutem sobre o aumento da manutenabilidade de sistemas COTS, enquanto que o capítulo 5 propõe um resumo das abordagens citadas e faz uma referência cruzada com os riscos apresentados. E finalmente, o capítulo 6 mostra as conclusões obtidas durante a pesquisa e futuros trabalhos.

Capítulo 2

Definições

Devido às muitas nuances encontradas sobre alguns dos termos utilizados nesta monografia, como por exemplo: manutenção, manutenabilidade, sistemas COTS, padrões de projeto procuramos neste capítulo clarificar estes termos para evitar diferentes interpretações.

2.1. Manutenção

Segundo [IEE90], a manutenção de um sistema é o processo de modificar um sistema de software ou um componente após a sua entrega, com o objetivo de corrigir falhas, melhorar o desempenho ou outros atributos, ou adaptar para uma mudança de ambiente.

Segundo Sommerville em [SOM03], Lientz and Swanson descobriram nos anos de 1980, que uma organização gastava no mínimo, 50% de todo o esforço de programação em manter sistemas existentes, e que Mckee afirma que entre 65% a 75% de todo esforço é gasto em manutenção.

As atividades de manutenção podem ser divididas em 3 categorias [SOM03]:

- Manutenção corretiva : Representam atividades direcionadas a corrigir defeitos encontrados no sistema como falhas em desempenho, funções, processamento ou mesmo de implementação.
- Manutenção adaptativa: Representam atividades que são feitas quando o sistema necessita de alterações estruturais, como mudança de banco de dados / sistema operacional / plataforma de desenvolvimento.
- Manutenção perfectiva: Representam atividades de implementação de novos requisitos no sistema.

Muitas vezes essas categorias não tem uma distinção clara entre elas. Por exemplo, ao encontrar falhas no sistema, a melhor alternativa talvez seja fazer adaptações (manutenção

adaptativa) ao invés de simplesmente corrigir a falha (manutenção corretiva). Essas interseções dificultam estatísticas sobre quais delas são mais realizadas, mas Lientz e Swanson afirmam que o esforço da manutenção pode ser dividido da seguinte maneira: 17% referem-se a manutenção corretiva, 18% a adaptativa e 65% a perfectiva [SOM03].

Neste sentido, estudar procedimentos que tendem a diminuir o esforço e facilitar as atividades de manutenção, tem um impacto direto sobre o custo total do sistema, beneficiando tanto a organização que fez o sistema, quanto o usuário que recebe um produto com menor custo e maior qualidade.

2.2.Processo de manutenção

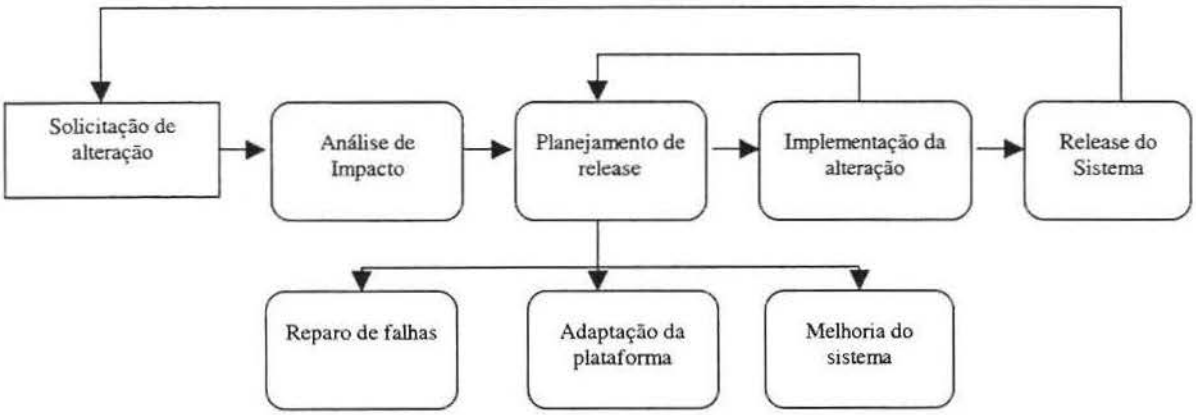

Figura 1 - Processo de Manutenção

O processo de manutenção geralmente é iniciado após a entrega do sistema ao cliente. A Figura 1 descreve um processo simples de manutenção que se inicia com uma solicitação de alteração do sistema. Em seguida, o custo e impacto dessas alterações são analisados para verificar como o sistema será alterado e estimativas serão feitas para avaliar o custo dessas alterações. Caso decida-se que as alterações sejam realmente incorporadas ao sistema, uma nova versão do sistema é planejada considerando-se todas as possíveis alterações (correção de erros, adaptação ou novas funcionalidades) no sistema. As alterações são então feitas, validadas e incorporadas ao sistema.

Apresentamos a seguir alguns problemas comuns à fase de manutenção de um sistema tradicional:

- Existe pouca visibilidade sobre o tamanho do software que será feita a manutenção, dificultando o planejamento e o cálculo de orçamentos. Os orçamentos referentes à manutenção são planejados com bases nos resultados dos anos anteriores que nem sempre estão atualizados (refletindo o custo efetivo), e não levam em conta as possíveis grandes mudanças necessárias, gerando orçamentos falhos e com grande desvio entre o custo previsto e o efetivo. [VIG97]

- Os testes geralmente não são planejados e quando realizados, não são documentados nem tampouco cobrem todas as funcionalidades do sistema COTS, gerando um sistema no qual não é possível descrever com confiabilidade quais requisitos foram implementados e testados e em que tipo de situações. A equipe de manutenção herda um sistema sem que haja conhecimento pleno do comportamento das funcionalidades do sistema. [CLA98]

- Métodos rudimentares e individuais de codificação, resultam muitas vezes em códigos ininteligíveis e de difícil rastreamento de erros. Esse problema pode ser constatado devido à falta de técnicas de engenharia de software ou quando não há um processo definido para produção de um sistema. [CLA98]

- Há falta de documentação atualizada que detalhe todos requisitos, descreva como foi planejamento do projeto, e os aspectos técnicos do sistema (especificações técnicas, arquitetura) dificultando o entendimento completo do sistema. Geralmente a documentação existente corresponde a uma versão inicial do sistema. mas que não foi atualizada conforme o andamento do projeto, e não serve portanto, como base para a equipe de manutenção. [VIG97]

- Há falta de um controle de versões que possibilite a identificação e recuperação rápida e confiável das versões entregues aos clientes. É muito comum encontar empresas que não sabem quais as funcionalidades contidas nas várias versões entregues aos clientes. [VIG97]

- Há falta de pontos de verificação para analisar se o que está sendo produzido está conforme os requisitos do cliente. É comum que as inconsistências entre o produto gerado e os requisitos do cliente sejam verificadas somente na fase final de testes. Isso resulta geralmente em um projeto com muitos erros que serão corrigidos pela equipe de manutenção.

- Há falta de métodos e procedimentos que assegurem um mínimo de qualidade do produto. [CHR03].

- Geralmente os analistas/engenheiros juniores são deslocados para as tarefas de manutenção, quando os mesmos não possuem prática em engenharia de software e tem pouco entendimento do processo de desenvolvimento e do produto como todo. [CLA98]

- Há pouca rastreabilidade entre os requisitos, código-fonte e seus produtos intermediários, impossibilitando que a equipe de manutenção reconheça quais códigos-fontes implementam um determinado requisito, muito menos os produtos intermediários produzidos. O resultado disso pode ser constatado na existência de código-fonte que nunca é executado, mas nem por isso é removido do produto, pois há relutância da equipe de manutenção em alterar o código existente com receio dos efeitos colaterais e perda de algumas funcionalidades. [CLA98]

- É comum encontrar equipes de manutenção e desenvolvimento utilizando códigos-fontes com diferentes versões, criando situações em que erros corrigidos em uma versão não são propagados para as versões mais atuais. [VIG97]

- A rapidez exigida pelos clientes para correção de erros, faz com que a equipe de manutenção crie verdadeiros "*band-aids*" no código-fonte sem nenhuma documentação das alterações e análise de impacto, dificultando ainda mais a própria manutenção, pois a equipe não tem mais controle sobre quais requisitos/erros foram implementados em determinada versão. [CLA98]

Ao analisar estes problemas encontramos algumas práticas que procuram minimizá-los:

- A gerência deve reconhecer a importância da equipe de manutenção, demonstrando que seu valor é o mesmo que o da equipe que desenvolveu o sistema [CLA98]
- Os melhores desenvolvedores devem ser desafiados e motivados a fazer parte da equipe de manutenção [CLA98]
- Criar um planejamento para manutenção adaptativa que permita a equipe de manutenção decidir quando determinadas partes do sistema deverão ser submetidas à reengenharia.
- Envolver a equipe de manutenção nas fases iniciais do projeto, desde a especificação até os testes finais [CLA98].
- Investir na definição processos para o desenvolvimento de software que garantam a padronização de métodos, documentação atualizada e gerência de configuração confiável [CHR03].

2.3.Gerência de configuração

Detalhamos a seguir os conceitos de versão, release e gerência de configuração, segundo Sommerville [SOM03]:

- Versão: Uma instância do sistema que difere de outras de alguma maneira. Novas versões do sistema podem ter diferentes funcionalidades, perfomance ou ter erros corrigidos. Outras versões podem ser funcionalmente equivalentes mas são executadas em plataformas diferentes [SOM03].
- Release : é a versão distribuída para clientes. Cada release deve necessariamente incluir novas funcionalidades, ou então, ser uma adaptação para hardwares diferentes [SOM03].
- Gerencia de Configuração: é o desenvolvimento e aplicação de padrões e procedimentos utilizados para gerenciar todas as versões resultantes de alguma correção de falha ou adaptação feitas no sistema.
  Os procedimentos de gerência de Configuração definem como armazenar e processar mudanças no sistema e exigem ferramentas que dêem suporte para armazenar as versões de produtos de sistema e possibilitem a reconstrução das versões através destes produtos.

2.4.Arquitetura de Projeto

Segundo Sommerville em [SOM03], a arquitetura de software envolve a descrição dos elementos a partir dos quais os sistemas são construídos, as iterações entre esses elementos, os padrões que orientam sua composição e as restrições sobre esses padrões.

A arquitetura de software tem como um dos principais objetivos : prover uma forte coesão entre módulos (módulos de código-fonte são fortemente integrados) e fraco acoplamento (módulos de código devem estar fracamente acoplados, com pouca ou nenhuma dependência entre módulos), além de maximizar o reuso, melhorar a manutenção e assegurar a portabilidade.

Para tanto são definidos os padrões arquiteturais, que expressam a idéia fundamental do esquema de organização do software e devem propor soluções eficientes e elegantes para problemas de arquitetura.

2.5.Padrões de Projeto

Segundo Gamma et al em [GAM95], um padrão é uma solução aplicada repetidamente, com pequenas variações, a problemas recorrentes em um dado domínio de conhecimento. São propostas de soluções eficientes e elegantes para problemas comuns de projeto de

software. Essas soluções são desenvolvidas e conhecidas por especialistas, e tornam-se padrões por serem reutilizadas várias vezes em vários projetos e por terem a sua eficácia comprovada.

Podemos citar as principais propriedades dos padrões como sendo:

- Capturam o conhecimento e a experiência de especialistas em projeto de software.
- Especificam abstrações que estão acima do nível de classes ou objetos isolados ou de componentes .
- Definem um vocabulário comum para a discussão de problemas e soluções de projeto.
- Facilitam a documentação e manutenção da arquitetura do software.
- Auxiliam a simplificar a complexidade da arquitetura de um projeto.
- O esforço extra gasto na fase de projeto é compensado pelos ganhos em flexibilidade e reuso.

Podemos citar como exemplos de padrões: façades, design patterns e mediators descritos a seguir.

O **façade** é um padrão que fornece uma interface única de acesso a um conjunto de interfaces, e geralmente não diminui o número de conexões. É recomendada sua utilização quando um conjunto de objetos se comunica de forma bem definida, mas de maneira complexa, resultado numa organização confusa, ou então, nos casos em que a reutilização de algum objeto é difícil devido às suas dependências em relação aos demais componentes do sistema.

A Figura 2 ilustra uma comunicação complexa onde os programas 1 e 2 têm acesso direto as interfaces dos produtos COTS do sistema.

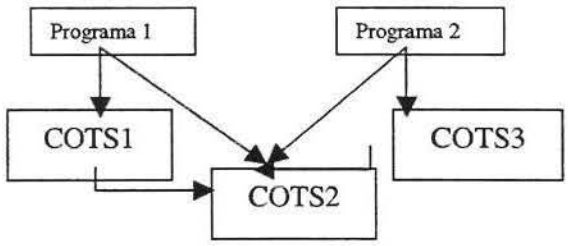

Figura 2 - Arquitetura complexa

Neste caso notamos alguns problemas: É necessário que cada programa tenha conhecimento prévio dos parâmetros exigidos por cada produto COTS, cada programa deve ter um tratamento de falhas para analisar as respostas provenientes do produto COTS e caso

o produto COTS em questão for removido ou trocado, todos os programas afetados devem ser recodificados.

A Figura 3, utiliza o padrão "façade" para reconstruir a arquitetura. Neste exemplo, os produtos COTS foram agrupados de modo que toda comunicação com eles seja feita somente através de uma interface façade , garantindo que os Programas 1 e 2 acessem os produtos COTS somente através da interface façade.

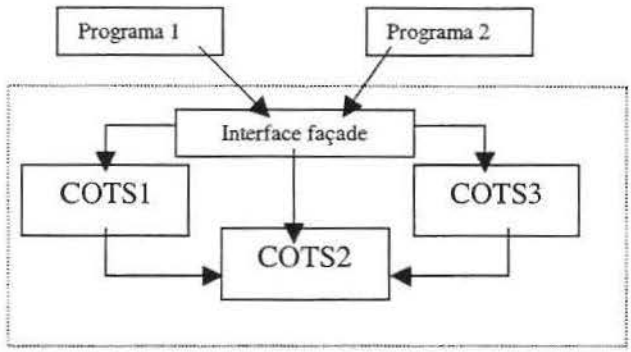

Figura 3 - Arquitetura com a utilização do padrão façade

Esta nova estruturação possui uma série de vantagens: possui um único controle de tratamento de falhas onde toda falha detectada entre o conjunto de produtos e cada programa pode ser facilmente identificado e documentado, restringe o acesso às interfaces dos produtos evitando acessos indevidos, diminui o número de componentes que comunicam diretamente entre si favorecendo a eventual troca de componentes, e facilita os testes pois cada programa pode ser testado como uma unidade simples antes de ser integrado ao sistema.

O **mediator** é um tipo de padrão que possui uma infraestrutura interna capaz de escolher e coordenar os componentes especializados necessário para manipular os dados em transição, possibilitando o suporte a várias alternativas de tratamento de dados [GAM95].

A Figura 4 exemplifica uma arquitetura em que vários produtos COTS de um mesmo sistema comunicam-se diretamente entre si.

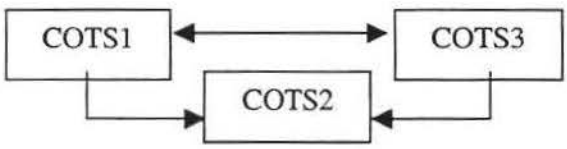

Figura 4 - Arquitetura sem mediator

Podemos verificar na figura acima os seguintes problemas arquiteturais: necessidade de conhecimento prévio dos parâmetros exigidos por cada interface, pode ser exigido neste caso que obrigatoriamente todos os produtos COTS tenham o mesmo tipo de interface (dificultando eventuais trocas de produtos), o tratamento de falhas é isolado, e caso seja necessária a troca de produtos há necessidade de reintegrar todo o sistema novamente.

Uma proposta para melhorar esta arquitetura é a utilização do padrão "mediator" que centraliza o tratamento das interações entre estes produtos como mostrado na Figura 5.

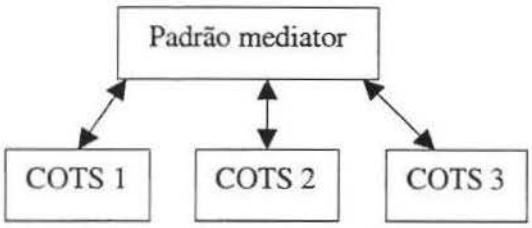

Figura 5 - Arquitetura com uso de mediator

Esta nova estruturação possui uma série de vantagens: um único controle sobre o tratamento de falhas onde toda falha detectada nos produtos COTS pode ser isolada e identificada , o acesso às interfaces de cada produto fica restrito evitando acessos indevidos, possibilita controlar os diferentes tipos de dados que são manipulados entre os produtos COTS favorecendo a eventual necessidade de trocar produtos COTS.

Um adaptador ou ***wrapper*** é um padrão de arquitetura inserido entre um componente e seu ambiente para controlar o fluxo de dados trocados entre eles. Ele funciona como uma "capa" que isola o produto COTS do sistema, controlando a entrada e a saída que cada produto recebe [GAM95].

É utilizado quando há conflito entre dois componentes do sistema e possibilita compatibilizar a comunicação entre os dois componentes problemáticos pois permite controlar o fluxo de entrada e saída que existe entre cada um destes componentes, situação comum no desenvolvimento de sistemas baseados em componentes pois sua implementação representa um baixo custo e permite controlar o comportamento do produto COTS no sistema, independentemente se a interface do produto é proprietária ou não.

Na Figura 6, ilustramos uma arquitetura que permite o acesso direto ao produto COTS, onde cada programa pode comunicar-se diretamente com as interfaces do produto COTS.

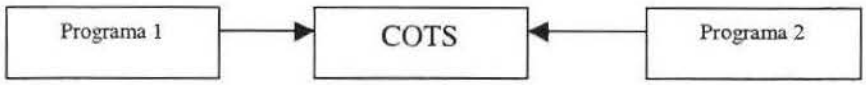

Figura 6 - Arquitetura sem wrappers

Na Figura 7, ilustramos a arquitetura da Figura 6 alterada para a inclusão de um *wrapper*, e notamos os seguintes benefícios : Não é necessário que cada programa tenha conhecimento prévio dos parâmetros exigidos pela interface do produto COTS, e sim somente dos parâmetros do wrapper, permitindo que tanto os programas quanto o produto COTS seja alterado sem prejuízo ao sistema; o tratamento de falhas é único e fica centralizado na interface criada pelo padrão wrapper, controlando todas as entrada e saídas.

Isso facilita a correção de falhas, monitoração do comportamento do produto, e principalmente permite a padronização de comunicação com o produto COTS. Essa padronização pode ser muito útil quando as interfaces do produto COTS são proprietárias e custosas de ser mantidas. Nesse caso, elas podem ser reprogramadas pelos padrões do tipo wrappers para serem interfaces padrões e de baixo custo de manutenção

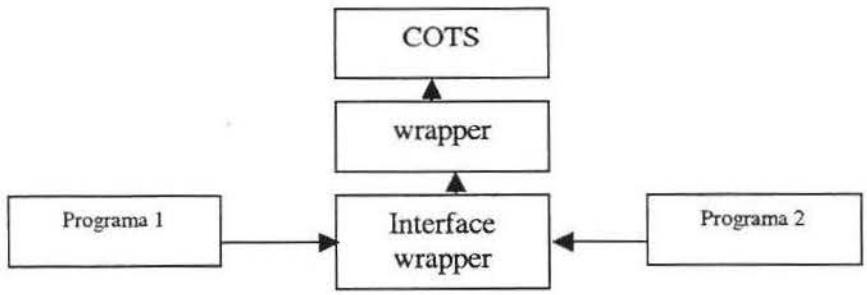

Figura 7 - Arquitetura com wrappers

2.6.COTS

Atualmente várias interpretações estão sendo dadas a palavra COTS. Ela tem sido usada para identificar "softwares de prateleiras" que podem representar softwares comprados em papelarias, bibliotecas desenvolvidas internamente e ate mesmo compiladores.

Utilizaremos neste estudo a definição utilizada por Browsword, Oberndorf e Sledge [BRO00], onde podemos definir um software *COTS (Commercial-off-the-shef)* pelas seguintes características:

1. É vendável e/ou licenciável para o público em geral
2. Oferecido por um fornecedor que tenta obter lucro dele
3. Desenvolvido e suportado pelo fornecedor que retém seus direitos intelectuais
4. Disponível em múltiplos e cópias idênticas
5. Usado sem modificação/acesso ao de código-fonte

Os uso de produtos COTS permite que a equipe de desenvolvimento de um sistema concentre-se apenas nos requisitos de alto nível do sistema, abstraindo-se assim, das funções de baixo nível que ficam sob responsabilidade dos produtos COTS. Além disso, muitos destes produtos COTS são desenvolvidos conforme padrões modernos e atualizados disponíveis no mercado, ofertados por mais de um fabricante, permitindo assim, uma escolha mais personalizada possibilitando a análise de requisitos e custo conforme a necessidade de cada sistema a ser desenvolvido.

Como o código-fonte não fica disponível, a maneira pela qual podemos acessar as funcionalidades dos produtos COTS fica sendo através de interfaces denominadas API (Application Programming Intefaces). Estas intefaces podem seguir padrões já existentes no mercado ou então ser do tipo proprietárias onde cada fabricante detém a patente sobre elas, e é através delas que os produtos COTS podem interagir com outros sistemas.

Consideramos como produtos COTS, os sistemas pré-construidos, vendidos ou licenciáveis em que os fabricantes não permitem acesso ao código-fonte que os produziu. Podemos citar como exemplo de produtos COTS: compiladores, sistemas operacionais, bibliotecas de funções fornecidas por terceiros, editores de texto, navegadores de internet.

2.7.Definição de Sistemas baseados em COTS

Segundo Wallnau [WAL98], um sistema baseado em COTS, aqui simplesmente denominado de *sistema COTS*, é aquele que integra vários produtos COTS em um mesmo sistema. Esses produtos COTS podem efetuar funções genéricas que são independentes do

domínio da aplicação, e geralmente fazem parte da infra-estrutura do sistema. Este tipo de sistema também é referenciado por alguns autores como COTS-Intensive Systems.

O termo *glue*, ***gluecode*** ou *glueware* é o nome que se dá ao código-fonte que permite a integração destes produtos COTS [VIG97]. A empresa que desenvolve o sistema COTS seleciona quais as funcionalidades a serem utilizadas tendo em vista os requisitos do sistema e define como elas serão utilizadas. O *gluecode* pode ser uma mistura de várias linguagens, inclusive a fornecida pelo fabricante do produto COTS, e caso não seja bem planejado pode assumir grandes proporções no sistema exigindo um grande esforço para desenvolvê-lo e mantê-lo.

Concluímos que formar um sistema baseado em COTS envolve tipicamente as seguintes atividades: identificar os produtos COTS a serem integrados, definir os fornecedores destes produtos, determinar as interfaces entre estes produtos e outros já existentes no sistema, adaptar os produtos, e construir o *gluecode* necessário (codificando funcionalidades, bibliotecas, scripts que não foram cobertos pelos produtos COTS).

Como foi dito no capítulo 1 de Introdução, iremos tratar nesta monografia de procedimentos para manutenção de sistemas que integram vários COTS, e não em sistemas que *desenvolvem* COTS, observando que neste último é possível o acesso ao código-fonte.

2.8. Definição de Manutenabilidade em Sistemas COTS

Trataremos durante todo esse trabalho referências que melhorem a manutenabilidade de um sistema COTS, buscamos então uma definição mais concisa para o termo manutenabilidade de um sistema COTS.

A definição da IEEE de 1990 para manutenabilidade é a seguinte : "A facilidade com que o sistema de software ou produto pode ser modificado para corrigir erros, melhorar o desempenho ou outros atributos, ou adaptar-se a ambientes modificados", mas como podemos definir, então, a manutenabilidade do ponto de vista de um sistema COTS ? Consideramos para tanto a definição de Mari et al em [MAR03] que define esse termo para um sistema baseado em componentes, uma vez que podemos considerar que cada produto COTS pode ser entendido como um "componente" do sistema COTS.

Mari et al em [MAR03] consideram que a manutenabilidade permeia todo o desenvolvimento de um sistema, e afirmam que as equipes de manutenção e desenvolvimento devem ter consciência de sua definição para que possam desenvolver sistemas que possuam esse requisito. Essa visão em relação às características da manutenabilidade, segundo os autores, permite que uma nova dimensão seja criada detalhando melhor a maneira pela qual as várias faces da manutenção podem ser afetadas.

As características (ou atributos) da manutenabilidade segundo os autores são:

- Flexibilidade - a facilidade em que um sistema ou produto pode ser modificado em aplicações ou em um ambiente diferente daquele em que foi especificamente desenhado.
- Modificabilidade - a habilidade para se fazer mudanças de maneira rápida e de custo efetivos e é subdividida em:
- extensibilidade - a habilidade de adquirir novos produtos
- portabilidade - a habilidade que o sistema têm de funcionar em diferentes hardwares, software ou em uma combinação dos dois.
- Reusabilidade - a habilidade da estrutura do sistema ou de algum de seus produtos de ser reutilizado em aplicações futuras.
- Integrabilidade - a habilidade de integrar produtos que foram desenvolvidos separadamente de modo que o sistema trabalhe corretamente.
- Testabilidade - a facilidade em que o software pode ser feito para mostrar suas falhas.

Essas características da manutenabilidade podem ser representadas:

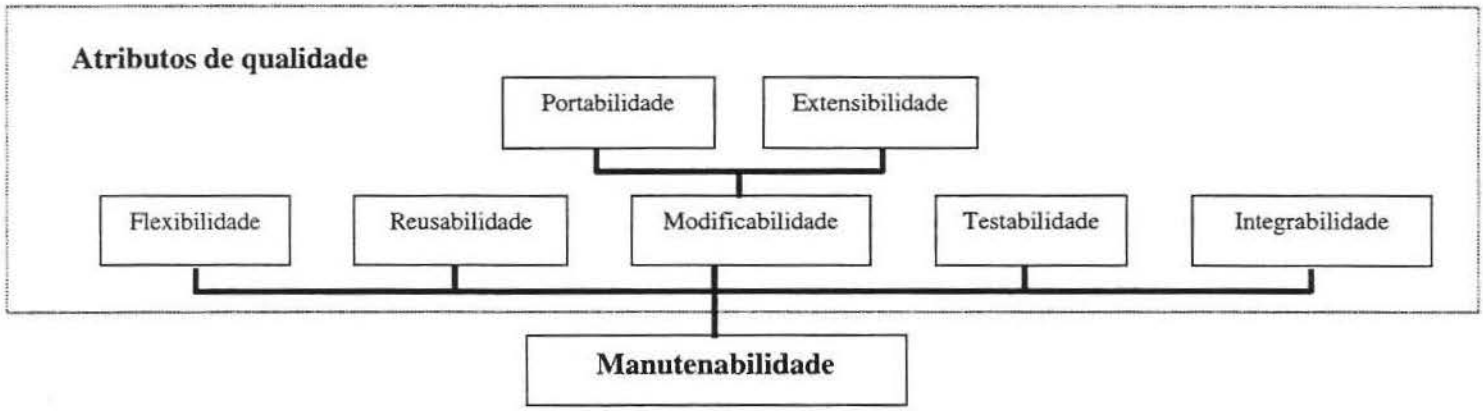

Figura 8 - Características manutenabilidade de um sistema COTS

Com esse conceito, os autores concluem que a manutenabilidade tem uma definição mais precisa quando ela é desmembrada em características, pois permite que os desenvolvedores de sistemas possam identificar quando, onde e como eles devem prestar atenção as diferentes faces da manutenabilidade, pois ela está presente em todas as fases do ciclo de vida de um sistema, variando conforme o nível de dimensão analisado.

Neste trabalho consideraremos esta subdivisão em características por concordar com os autores sobre as vantagens dela, além de possibilitar a avaliação de cada característica da manutenabilidade isoladamente das outras, favorecendo a identificação de estratégias que as melhorem.

Capítulo 3

Riscos associados à manutenção de Sistemas COTS

Clapp e Taub [CLA98] afirmam que a manutenção de sistemas COTS requer algumas mudanças nas atividades da manutenção tradicional além de apresentar riscos diferentes daqueles dos sistemas tradicionais porque tanto a equipe de desenvolvimento quanto à equipe de manutenção devem manipular verdadeiras "caixas-pretas" pois o acesso ao código-fontes dos produtos COTS não é permitido. Frente a estas diferentes atividades e perspectivas, o paradigma de construir sistemas COTS nos leva a reavaliar os riscos que possam afetar a manutenção de um sistema COTS

O principal objetivo deste capítulo consiste em conscientizar a equipe do projeto sobre os riscos associados a sistemas COTS, permitindo que haja uma gerência sobre eles, onde se torne possível decidir quais e quando eles serão tratados, evitando dessa forma que eles ocorram inesperadamente no projeto.

Porém é importante ressaltar que estes riscos devem ser gerenciados desde o desenvolvimento do sistema, e não somente na fase de manutenção, o que nos levou a agrupá-los em: requisitos, planejamento, arquitetura, controle de qualidade e gerência de configuração, indicando o momento do projeto em que eles geralmente ocorrem.

Após o levantamento destes principais riscos, discutiremos nos capítulos 4 e 5 , quais as atividades que poderão ser realizadas para mitigar/conter estes riscos. Assim sendo cada um destes riscos possui uma identificação única para que possam ser referenciados posteriormente.

Devemos ressaltar que este trabalho não visa definir métodos para gerenciamento de riscos, mas sim servir de base para que eles possam ser feitos.

3.1. Requisitos

3.1.1. Há falta de aplicação de técnicas específicas para seleção dos produtos COTS a serem utilizados no sistema, o que gera incertezas em relação à adequação e viabilidade do produto a ser incorporado. Há casos em que a escolha é feita somente através da análise da

documentação, sem que haja testes prévios que assegurem os requisitos do sistema estejam satisfeitos pelos produtos COTS selecionados.
A má escolha de um produto ou de um conjunto pode comprometer toda a qualidade, desempenho e compatibilidade do sistema, dificultando sua manutenção. [VIG97][KON96].

3.1.2. Há falta de documentação e acompanhamento dos riscos que estão relacionados à seleção de um determinado produto COTS. Dificilmente um produto de prateleira satisfaz todos os requisitos funcionais e não-funcionais (portabilidade, usabilidade), e neste caso é essencial identificar quais os riscos que a organização deve acompanhar para que eles não impactem de maneira negativa no projeto.

3.1.3. O controle sobre os requisitos de cada produto COTS está nas mãos dos seus respectivos fabricantes, ou seja, adicionar e/ou remover requisitos depende do planejamento do fabricante, pois somente ele detém o acesso ao código-fonte. Caso a equipe de manutenção detecte erros no produto, cabe ao fabricante decidir ou não pela correção e quando ela será incorporada ao produto COTS [VIG97].

3.1.4. Uma nova versão de um produto COTS pode ou não preencher todos os requisitos das versões anteriores, ou mesmo, possuir funcionalidades superiores ou inferiores às desejadas. Caso eles sejam instalados no sistema COTS já em uso, esse aumento ou diminuição de requisitos pode refletir no sistema de modo negativo. Isto é, o sistema pode ficar com muito mais funcionalidades que o desejado, ou então, perder algumas fundamentais alterando desse modo, os requisitos de todo o sistema COTS [VIG97]

3.1.5. Em sistemas que se fazem necessários à segurança e a integridade dos dados, podem estar embutidos vírus do tipo "cavalos de tróia" (vírus que são ativados quando o produto for instalado), uma vez que o código-fonte não pode ser verificado. [CHR98]

3.1.6. Os produtos COTS que integram um mesmo sistema COTS podem oferecer níveis de segurança diferentes, obrigando que seja utilizado o menor denominador em comum. Por exemplo, um determinado produto COTS pode oferecer níveis de segurança detalhados enquanto outro permite todo e qualquer acesso ao sistema.[CHR98]

3.1.7. Há pouca ou nenhuma ligação entre os requisitos do sistema e os produtos COTS que os satisfaçam. Caso algum requisito seja alterado ou removido durante o projeto, a equipe de projeto tem duvidas em relação a manter, alterar ou retirar os produtos COTS.

3.1.8. Há versões de produtos COTS que apresentam restrições geográficas, especialmente aqueles que manipulam algoritmos criptográficos. Isso pode restringir seu uso e dificultar a tarefa de substituí-lo quando for necessário. [VIG97]

3.2.Planejamento

3.2.1. Há falta de definição de critérios para realizar troca de versões dos produtos utilizados no sistema COTS, dificultando a manutenção dos mesmos. Isso acontece porque os fabricantes de produtos COTS podem lançar versões novas nem sempre compatíveis com a antiga, com requisitos diferentes da anterior, e geralmente estes lançamentos são assíncronos e independentes dos lançamentos dos outros produtos COTS que compõe o sistema.

3.2.2. Há possibilidade de existir incompatibilidades entre produtos do mesmo sistema COTS. Ao realizar a troca de um produto COTS do sistema, podem ocorrer alguns problemas referentes à nova versão: incompatibilidade com o restante dos produtos COTS do sistema, exigência da instalação de novas versões dos outros produtos COTS, recodificação de parte do sistema pois a nova versão pode exigir formatos de dados diferentes (mudanças nos arquivos e banco de dados existentes), incompatibilidade com a versão do hardware existente, retreinamento da equipe, e até mesmo, alteração no desempenho e nos requisitos do sistema.[CLA98]
Por outro lado, devem ser verificas as conseqüências de não realizar a troca um determinado produto COTS do sistema. Clapp e Taub [CLA98] citam que nestes casos, pode ocorrer a perda do suporte do vendedor para as versões que já estão instaladas e em uso, ou então, a impossibilidade de comprar novas cópias ou obter licenças adicionais para a versão que está em uso atual.

3.2.3. Há falta de planejamento do custo de um produto COTS. Alguns itens que tipicamente envolvem sua compra não estão previstos nos orçamentos do projeto como: renovações de licenças, o (re)trabalho que existe quando há alteração dos sistemas resultantes de novos releases e versões, (re)treinamento específico em tecnologias utilizadas nos produtos e (re)teste de todo o sistema [CLA98].

3.2.4. Há falta de treinamento para manipulação dos produtos COTS, uma vez que eles podem exigir treinamento de novas tecnologias e compreensão profunda de todas funcionalidades do produto que nem sempre são tarefas triviais. O que ocorre na maioria das vezes é que pessoas sem treinamento adequado integram sistemas, sem ter conhecimento de todas as funcionalidades, definindo arquiteturas muito complexas e de difícil manutenção [LAW01].

3.2.5. Há pouca utilização de técnicas para gerenciar riscos ou mesmo analisar os riscos referentes a cada produto COTS e a integração deles. Vale ressaltar que a identificação e a avaliação dos riscos no início do desenvolvimento não é suficiente. Deve haver um

acompanhamento contínuo destes riscos durante todo o ciclo de vida do sistema uma vez que a integração dos produtos é feita em etapas e não em uma única vez, podendo gerar situações com novos riscos, inclusive na fase de manutenção [LAW01][CHR03].

3.2.6. Há pouca definição das estratégias de relacionamento com o fabricante dos produtos. Faltam definições e documentação claras sobre quais tipos de licenças adquiridas, informações se houve desconto sobre o volume de produtos comprados, tipo de suporte técnico adequado, treinamento necessário para utilizar os produtos e detalhamento sobre os conteúdos e erros conhecidos das novas versões. [LAW01]

3.3. Implementação

3.3.1. Atualmente poucos fabricantes de produtos COTS disponibilizam acesso à análise de desempenho e a detecção de falhas que possibilitem que as equipes de desenvolvimento e manutenção possam rastrear e monitorar os erros encontrados no sistema e atribuí-los aos produtos COTS. [HIS98]

3.3.2. Há pouca utilização de técnicas e arquiteturas mais apropriadas para o desenvolvimento de sistemas COTS. O que normalmente ocorre é que as empresas relegam a tarefa de construção da arquitetura para os programadores, permitindo que cada um decida pelo que melhor que lhe convier e altere essa arquitetura quando achar necessário, motivados pela visão restrita da funcionalidade que cada um irá desenvolver. O resultado disso é a falta de uma arquitetura que contemple o sistema como um todo, onde cada módulo do sistema é conhecido somente pelo programador que o desenvolveu, dificultando as alterações feitas pela equipe de manutenção. [KON96]

3.3.3. Há falta de controle nas alterações no código-fonte: Geralmente o cliente tem acesso direto ao programador e exige alterações no sistema. Elas são feitas sem haja a devida alteração (pois nem sempre as alterações fazem parte dos requisitos do sistema), análise do impacto destas alterações (custo, impacto sobre outros produtos COTS, arquitetura e desempenho) e atualização da documentação.

Sem saber os reais motivos pelos quais as alterações foram feitas, a equipe de manutenção muitas vezes continua com versões problemáticas e com pouca possibilidade de alterar o código pois não sabe os efeitos colaterais das mudanças, ou mesmo quais requisitos serão afetados. [LIN98]

3.3.4. Há pouco planejamento sobre o impacto de alterações no ambiente do usuário do sistema COTS. Há casos de incompatibilidade entre a nova versão do produto COTS com o

hardware do usuário, ou vice-versa, quando o usuário alterar o hardware o sistema COTS pode ser incompatível. [BRO00]

3.3.5. Há falta de planejamento ao integrar componentes do sistema COTS. Ao integrar os vários produtos COTS de um sistema COTS, podem ocorrer falhas nesta integração que podem estar em dois pontos: ou nos produtos COTS que foram integrados e/ou na interação entre alguns destes produtos COTS. Isso acontece particularmente quando não se tem controle sobre as entradas e saídas dos produtos COTS ou então, quando fatores como o sistema operacional, gluecode e instalação de releases alteram o comportamento de produtos COTS, Nestes casos, mesmo que seja identificado o problema nem sempre o fabricante do produto se interessa em resolvê-lo, tornando a equipe de desenvolvimento/manutenção refém destas correções.

3.3.6. Há problemas de incompatibilidade entre uma nova versão disponível no mercado de um produto COTS e o restante do sistema COTS que o engloba, sendo necessária uma atualização parcial ou total do sistema. Esse problema fica mais grave quando a empresa for obrigada a sempre realizar alterações para continuar a receber suporte do fabricante do produto COTS. [CLA98][BRO00]

3.3.7. Há excesso de retrabalho de código quando uma nova versão do produto COTS exigir novos padrões de dados como: banco de dados diferente, tamanho alterado de tabelas, parâmetros de funções diferenciados, linguagens diferentes, sem mencionar os novos treinamentos necessários que estão vinculados a estas mudanças.

3.3.8. Há falta de documentação atualizada e completa do produto COTS. Informações do tipo: "tratamento e correção de falhas", detalhamentos completos de interfaces, listagens de erros encontrados (e se foram ou não corrigidos) e incompatibilidades de cada produto são difíceis de serem encontradas atualizadas.

3.4. Manutenção

3.4.1. Há demora em responder ao cliente sobre os erros identificados no sistema COTS. Essa demora pode ser maior ainda se o produto COTS não for desenvolvido no Brasil, como é a maioria dos casos, impactando no tempo que a equipe de manutenção tem para resolver problemas. Em outros casos, o erro não será corrigido pelo fabricante do produto COTS, e portanto, o sistema COTS permanecerá com o erro frustrando o cliente.

3.4.2. Há falta de gerenciamento sobre o contrato de manutenção, gerando situações como descontinuidade do suporte ou ausência do fornecedor do mercado sem prévio aviso [CLA98].

3.4.3. Há falta de consistência nos testes para corrigir os erros encontrados pelo cliente. As equipes de manutenção geralmente não simulam os ambientes personalizados do cliente em que os erros foram detectados, impossibilitando a identificação clara do problema [HIS98].

3.4.4. Há muita dificuldade alterar o *gluecode* em sistemas COTS. A documentação técnica do sistema é precária e desatualizada, dificultando a previsão do comportamento do sistema. Identificar e corrigir corretamente falhas ficam na dependência das pessoas que participaram do desenvolvimento do sistema, ou de especialistas com larga experiência na área [HIS98].

3.5. Controle de Qualidade

3.5.1. Há falta de uma padronização de qualidade de produto COTS. O controle na qualidade de cada produto COTS depende da visão que seu fabricante tem sobre qualidade. Para alguns deles, a competição para lançar rapidamente um novo produto pode diminuir a qualidade do produto. [CLA98]

3.5.2. As falhas contidas são corrigidas caso e quando o fornecedor desejar consertá-las, pois disponibilidade e qualidade da documentação, treinamento, consultoria e suporte estão nas suas mãos. [CLA98]

3.6. Gerência de Configuração

3.6.1. Não há registro das diferenças entre as várias versões de um mesmo sistema COTS. Algumas empresas vendem sistemas COTS como itens de prateleira e paralelamente fornecem personalizações destes sistemas COTS, emitindo novas versões do sistema COTS. Essas emissões são feitas a esmo sem que haja um controle sobre o conteúdo de cada versão emitida. Como conseqüência disso, a equipe de manutenção não tem mais controle sobre qual versão dos produtos COTS foi instalada em qual cliente, e portanto não sabe qual delas deve ser restaurada para corrigir eventuais falhas que surgirem. Esse problema assume proporções maiores caso seja necessária à alteração do *gluecode* [CLA98][LIN98][KON96].

3.6.2. Não há registro sobre as versões das ferramentas utilizadas durante o desenvolvimento do sistema COTS. Ao efetuar alterações no código, a equipe de manutenção muitas vezes utiliza versões de compiladores, bibliotecas e ferramentas diferentes das que foram utilizadas durante o desenvolvimento para reconstruir o sistema, gerando uma série de incompatibilidades no sistema [HIS98].

Capítulo 4

Revisão da literatura

Muitos autores, como Carney et al em [CAR00], Vidger em [VIG97] e Clapp [CLA98], concordam com a afirmação de que a manutenabilidade não é algo que possa ser adicionado simplesmente ao sistema, mas sim deve ser considerada desde os primeiros estágios do desenvolvimento do sistema.

Com esta abordagem e tendo em vista os riscos levantados no capítulo anterior, procuramos na literatura diferentes estratégias relacionadas às várias etapas do desenvolvimento de um sistema COTS que pudessem melhorar a fase de manutenção de um sistema COTS.

É importante ressaltar que evitamos buscar um processo específico de manutenção, uma vez que em projetos mal estruturados, a manutenção pode até ser inviabilizada segundo Vigder [VIG97], e nesses casos este estudo não teria aplicabilidade.

4.1. Guia de gerenciamento para manutenção de sistemas COTS

Clapp e Taub apresentam em [CLA98] um "guia de gerenciamento para manutenção de sistemas COTS", contendo recomendações que, uma vez seguidas durante o planejamento e a implementação do sistema COTS, irão impactar diretamente na manutenção do sistema. Além disso, o trabalho também discrimina as atividades da manutenção de sistemas tradicionais que devem ser alteradas e quais novas atividades devem ser feitas na manutenção devido ao uso de produtos COTS. A seguir apresentamos um resumo destas atividades.

1. Realizar análise e pesquisa de Mercado

A pesquisa de mercado deve ser feita para determinar a disponibilidade de novos hardware e softwares disponíveis, e também para estimar em quanto tempo ou quais tipos de mudanças deverão ser feitas nos produtos COTS que estão sendo usados atualmente no sistema COTS. Essa informação é utilizada para fornecer uma análise de impacto para eventuais trocas de produtos dentro do sistema COTS.

Ela também pode ser útil caso a organização tenha interesse em antecipar-se das mudanças dos produtos COTS, permitindo estimar quão cedo e quais tipos de mudanças deverão ser feitas aos produtos COTS que já estão integrados no sistema COTS, tornando possível avaliar e encontrar produtos equivalentes ou até mesmo melhores e que podem substituir os utilizados atualmente. A autora ressalta que essa pesquisa de mercado deve ser um processo contínuo, haja vista a alta taxa em que novos produtos entram e saem do mercado.

2. Gerenciar as licenças utilizadas

A utilização de produtos COTS implica na compra de licenças para sua utilização, e envolvem tarefas que afetam diretamente o custo e a flexibilidade do sistema COTS e devem ser, portanto, planejadas. Os autores citam como exemplos de tarefas relacionadas às licenças: quem deverá comprá-las, a quem elas pertencerão, quem irá mantê-las, sob quais condições que elas serão utilizadas (licença de uso e avaliação), quais versões serão utilizadas, qual tipo de hardware, o tempo de uso, número de cópias, tipo de serviço do suporte ao cliente, nível de configuração do produto, quantidade de versões para desenvolvimento e execução.

Estas atividades devem estar documentadas para que a equipe de manutenção tenha acesso a todos os acordos feitos sobre as licenças compradas.

3. Realizar análise de alteração de versão

Essa atividade irá determinar o custo e a dificuldade de se introduzir uma nova versão de um produto COTS no sistema, seja ele um produto COTS novo ou uma nova versão do que está sendo atualmente utilizado.

Esta tarefa envolve também a análise do impacto de não efetuar uma determinada troca de versão, analisando os benefícios em potenciais contra os custo relacionados a tempo. Isso porque um o trabalho adicional para acomodar todas as mudanças necessárias em alguns casos pode ser tão contra-produtivo que superaria os possíveis benefícios.

Os autores ressaltam que não realizar estas trocas de versões pode significar na perda do suporte do produto, dependendo do acordo feito com o fabricante, e caso alguma alteração seja necessária no futuro, a empresa devera absorver o esforço de ajustar o sistema podendo

exigir um custo mais alto do que fazer a troca. Além disso, caso a empresa opte por realizar a troca após longo tempo o resultado desta estratégia pode resultar em custo alto, porque alguns fabricantes cobram por todos as versões intermediárias lançadas.

3. Programa de planejamento e cronograma

O planejamento do custo de uma manutenção de um sistema COTS deve ser previsível e devem ser levados em conta os seguintes itens: custo das renovações de licenças, necessidade de retestar o sistema, necessidade de retreinamento da equipe e custo das atividades de pesquisa de mercado, análise de impacto e gerenciamento de licenças.
Os autores sugerem algumas estratégias para controlar os custos:

- Planejar ciclos regulares de releases dos produtos COTS que durem de 2 a 3 anos (ou em menos tempo em casos excepcionais), pode levar a custosas e difíceis adaptações no sistema COTS, porém diminuem o custo com manutenção.
- Diminuir o número de fabricantes com que se trabalha gerando relacionamento mais estreito, diminui muitas vezes o gasto e a quantidade de conflitos entre produtos COTS.
- Sincronizar a instalação de novas versões dos produtos COTS com a instalação de novas versões do sistema
- Planejar a troca de hardware em ciclos regulares (por exemplo: A cada três ou quatro anos) para evitar que fiquem obsoletos e que aceitem com facilidade a instalação de novas versões.

4. Controle de qualidade

O controle de qualidade aqui mencionado refere-se a manter o comportamento correto do sistema, mantendo-o disponível. Os autores recomendam manter um acordo com o fabricante do produto, geralmente através de licenças, para determinar a correção dos eventuais problemas.

Os autores ressaltam que uma das grandes vantagens dos produtos COTS é a grande quantidade de usuários ao redor do mundo que ajudam a encontrar problemas no produto de modo que a qualidade do mesmo tende a aumentar. É muito comum o fabricante do produto COTS, periodicamente agrupar a resolução destes problemas e enviar aos usuários do produto uma versão contendo as correções. Nestes casos, a equipe de manutenção do sistema COTS deve estar apta a decidir se essa nova versão deve ou não ser instalado, tendo em vista que, os erros corrigidos nela podem afetar gravemente o sistema, e definir quais as partes de sistema devem ser retestadas para manter a qualidade atual do sistema COTS.

Além disso, outro item importante associado à qualidade, é o modo pelo qual os problemas encontrados no sistema COTS são identificados, armazenados e analisados. Deve ser possível identificar quando um problema foi originado por um produto COTS e quanto

tempo levará sua correção. O tipo de informação dado pelo fornecedor define a confiabilidade de cada produto, e pode ser útil quando for necessário determinar se um produto COTS deve ser trocado por suas funcionalidades ou por seu fornecedor.

5. Gerência de Configuração

No caso de sistemas COTS, os autores recomendam que sejam registradas para cada produto COTS que compõe o sistema, informações do tipo: quais versões dos produtos COTS estão instaladas em cada cliente, qual tipo de plataforma e qual o tipo de manutenção a ser dada no cliente. Ressaltamos neste trabalho a importância de também armazenar informações do tipo: quais erros foram encontrados e corrigidos, quais erros encontrados ainda não foram corrigidos, quais versões foram substituídas e quais funcionalidades do produto atendem aos requisitos do sistema COTS.

Esse conjunto de informações permite que as correções de erros e as novas versões do sistema COTS sejam distribuídas corretamente, além de permitir a reconstrução de sistemas que foram total ou parcialmente destruídos.

6. Plano de Gerenciamento do Ciclo de Vida do Produto COTS

Os autores além de citar estas atividades, recomendam a elaboração de um Plano de Gerenciamento do Ciclo de Vida do produto COTS. Esse plano contém guias gerais e decisões que devem ser feitas durante os estágios de planejamento e implementação que afetem a manutenção dos produtos COTS. O plano engloba o planejamento de todas as atividades de manutenção novas e alteradas citadas acima, além dos seguintes itens :

- Realizar uma análise de restrições : verificar políticas, regras, regulamentações e quaisquer decisões que sejam impostas ao sistema, documentado-as para análise posterior.

- Realizar análise de requisitos: Todos os procedimentos pelos quais determinados produtos COTS foram preteridos devem estar documentados no plano, inclusive o(s) método(s) de escolha e seu(s) respectivo(s) resultado(s)

4.2.Métodos para seleção e avaliação de produtos COTS

Compreendendo que a seleção correta dos produtos COTS que integrarão o sistema definirão o sucesso da arquitetura e manutenção do sistema, buscamos alguns métodos disponíveis na literatura e apresentamos então, um breve resumo de alguns.

1. Fase de avaliação de produtos do modelo CISD

O modelo CISD, COTS-Based Integrated System Develoment, desenvolvido por Tran e Liu em [TRA97] generaliza o processo de selecionar, avaliar e integrar produtos COTS. Ele é composto por três fases seqüenciais: Identificação do Produto, Avaliação do Produto e Integração do Produto.

Inicialmente é feita uma lista com os produtos COTS mais compatíveis com os requisitos. Essa lista é ordenada por algum critério definido pela empresa (custo, facilidade de operação, plataforma). É iniciada então, a fase de avaliação que trata de criar um protótipo do software para uma integração temporária.

Os protótipos passam então por 3 etapas seqüenciais de teste : funcionalidade, interoperabilidade e desempenho. Na primeira, é verificada se as funcionalidades oferecidas pelo produto satisfazem os requisitos solicitados pela organização. Na segunda, verifica-se o comportamento do produto depois de ter sido integrado aos outros produtos do sistema (*gluecode* e produtos COTS). Na terceira e ultima fase, é feita a análise de desempenho que geralmente fica por último por representar um requisito do sistema como um todo, e consiste basicamente, em uma análise quantitativa do efeito do produto COTS sobre o desempenho do sistema.

Mas uma restrição deste método é a necessidade de obter cópias dos produtos COTS avaliados, além de envolver prazos para realização dos testes. Isso muitas vezes não é viável para a empresa e nestes casos, os autores recomendam alguns métodos para avaliação baseada em restrição de tempo e custo como por exemplo, o Comprehensive Evaluation (CE) e o First-Fit Evaluation (FE). O primeiro considera que todos os produtos sejam testados, e o produto a ser selecionado é aquele obtiver melhores resultados em todos os testes, enquanto que o segundo interrompe o processo de avaliação assim que um produto satisfizer todos os testes, exigindo que os requisitos estejam os mais completos possíveis, para que a avaliação seja bem sucedida. O aspecto que finaliza a etapa de avaliação, consiste em considerar atividades como treinamento, custo e capacidade do vendedor

2. Off-the-Shelf-Option (OTSO)

Um dos primeiros métodos que surgiram para avaliar produtos COTS, foi proposto por Kontio et al em [KON96] e [KON96a], e é subdivido nas seguintes fases: localização, avaliação e seleção de produtos reusáveis, e fornecer técnicas específicas para definir alguns critérios de avaliação. Este critério de avaliação, por sua vez, é subdividido em: requisitos, características de qualidade do produto, compatibilidade de domínio e arquitetura e interesses estratégicos.

Para efetuar a análise dos resultados, é feita uma análise dos custos para adquirir cada produto COTS e compara custo e benefícios de cada alternativa, e utiliza a técnica AHP

(Analytic Hirerchy Process) [SAA90] para consolidar os resultados da avaliação e suportar o processo de tomada da decisão.

O ponto falho deste método é que ele se preocupa em definir os critérios de avaliação, porém não oferece orientação de como adquirir e modelar os requisitos.

3. Requirements-driven COTS product evaluation process (RCPEP)

Lawlis et al em [LAW01] sugerem um processo formal para avaliar produtos COTS, o RCPEP, que consiste em identificar produtos COTS, usar critérios para separar os produtos mais potenciais e utilizar cenários de testes de desempenho para avaliar os produtos.

O método propõe quebrar os requisitos para torná-los mais simples, dar peso a cada um deles, e então dar notas para os produtos possíveis, resultando em um valor que define a aderência do produto em relação aos requisitos. Uma matriz é montada para gerenciar esses dados, e o processo se repete até chegarmos em um conjunto de produtos que integrará o sistema.

Neste método, algumas exigências pelos autores: a mesma equipe deve avaliar cada produto, os produtos COTS devem ter configuração semelhante, toda avaliação deve usar os mesmos cenários de testes e dados de entrada, e os avaliadores devem aplicar os mesmos requisitos e critérios para todos nos produtos.

4. Método CRE (COTS based Requirements Engineering)

Ele é fundamentado no processo iterativo de aquisição de requisitos e seleção/rejeição de produto COTS. No início do processo há grande quantidade de produtos e poucos requisitos, e à medida que ele evolui, o quadro é invertido: há poucos produtos COTS e grande quantidade de requisitos. Em seguida são utilizados métodos de avaliação exaustiva pois os produtos são examinados detalhamente em relação aos requisitos e a autora sugere a utilização de algumas técnicas de tomada de decisão para fazer a escolha por um produto final.

Alves em [ALV01] propõe um método orientado a requisitos que tem como uma das principais estratégias enfocar a descrição dos requisitos não-funcionais durante o processo seletivo, uma grande diferenciação deste método para os citados anteriormente.

4.3.Atividades para manutenção de Sistemas COTS segundo Vigder

Vigder em [VIG98] afirma que a equipe de desenvolvimento de sistemas COTS pode construir um sistema que garantam manutenabilidade das seguintes maneiras: Identificando todas as atividades particulares a gerência e manutenção de um sistema COTS, Identificando características técnicas de produtos COTS que tornam o sistema manutenível e selecionar

produtos conforme esses critérios e identificando uma arquitetura que integre os produtos COTS e selecionar produtos COTS que satisfaçam essa arquitetura.

A seguir, apresentamos o ponto de vista do autor para cada uma das maneiras citadas acima.

1. Atividades associadas à gerência e à manutenção de um sistema COTS

O autor recomenda que a equipe de manutenção realize para cada alteração necessária um ciclo de tarefas que incluem: avaliação, integração e testes. A avaliação é necessária para que haja uma análise do impacto em termos de desempenho, uso de recursos e análise da compatibilidade do *gluecode* com a nova versão dos produtos.

Outra atividade que deve ser feita é a monitoração do sistema COTS, onde seja registrado o comportamento do sistema, permitindo que a equipe de manutenção tenha dados para melhorar o desempenho do sistema, observar anomalias no comportamento e conduzir análises de falhas do sistema.

Além destas atividades, a Configuração do Sistema é diferenciada dos projetos tradicionais pois envolve o gerenciamento de produtos e não mais de códigos-fonte, devendo a equipe de manutenção realizar as seguintes atividades : acompanhar versões de produtos COTS disponíveis, efetuar a gerência de configuração do *gluecode* dos produtos que estão sendo desenvolvidos e mantidos, acompanhar o histórico da configuração de instalação de cada produto, determinar versões compatíveis com os outros produtos COTS e gerenciar licença e suporte para cada produto

2. Métodos para garantir seleção de produtos COTS que promovam a manutenabilidade de sistemas COTS

Ainda segundo o autor, outra maneira para construir um sistema manutenível, é selecionar produtos COTS que facilitem a manutenção. O uso bem sucedido de produtos COTS, requer que a seleção e avaliação deles, sejam realizadas no início das etapas de desenvolvimento em conjunto com atividades de engenharia de requisitos e design de alto nível da arquitetura.

O autor cita que algumas características técnicas dos produtos COTS devem ser observadas na seleção e podem facilitar a manutenção :

- capacidade de customização do produto: é a habilidade do produto ser alterado para satisfazer um requisito ou um conjunto deles. Apesar do código-fonte não estar disponível, alguns fabricantes permitem a modificação da configuração e/ou do comportamento do produto COTS através de interfaces API (Application Programming

Interface), linguagens *scripts*, arquivos de configuração e mecanismos de herança em que podem ser codificados para herdar e especializar comportamentos dos objetos do produto;

- Funcionalidades exportáveis : identificar quais as funcionalidades do produto COTS podem ser programadas através da sua interface. Isso define como o produto será integrado no sistema COTS, possibilitando o controle dos dados de entrada e saída de cada produto COTS. Esse controle favorece o encapsulamento de funcionalidades, inserção de pontos para análise de desempenho e testes, e a utilização de padrões de projeto (ver cap 2).
- Visibilidade interna : habilidade do produto COTS de permitir o acesso ao seu comportamento interno durante tempo de execução. Por exemplo: emitem relatórios de comportamento e desempenho, monitoram a utilização de recursos utilizados pelo produto. Essa é uma habilidade que facilita detectar se um erro identificado no sistema COTS foi originado por algum dos produtos COTS que integram o sistema ou pelo *gluecode*.

3. Métodos para garantir manutenabilidade de sistemas COTS através de princípios de arquitetura e design

A terceira maneira para construir um sistema COTS manutenível segundo o autor, é determinar durante o desenvolvimento do sistema COTS uma arquitetura que favoreça a manutenção do mesmo. Isso pode ser feito se o *gluecode* for utilizado como a fundação da arquitetura, na qual os produtos COTS são inseridos ou removidos, permitindo assim que a equipe de manutenção possua controle máximo da arquitetura. O autor sugere alguns critérios que devem ser observados na construção do *gluecode*:

- Tratamento de falhas

Uma das características importantes de um sistema é a maneira pela qual podemos detectar e manipular as falhas encontradas. Independentemente se o produto fornece ou não uma maneira adequada para tratar falhas, é importante que a equipe de manutenção tenha maneiras consistentes e eficazes de detectar e manipular as falhas, ou seja, é necessário identificar aonde elas ocorreram, constatar sua origem e minimizar o impacto delas no sistema.

- Visibilidade entre produtos

A equipe de manutenção deve estar apta não somente a observar o comportamento de cada produto COTS, mas também seus inter-relacionamentos. Caso os produtos COTS que integram o sistema não permitam nenhum tipo de visibilidade interna, podem ser utilizados programas que monitoram alguns dos seguintes itens: canais de comunicação (como

Sniffers), desempenho, utilização dos recursos de hardware e forneçam estatísticas de tempo de processamento.

- Customização do sistema

A arquitetura do sistema deve ser flexível de modo a aceitar facilmente alterações, uma vez que um sistema COTS pode ser alterado por vários motivos como troca de requisitos, incorporação de novas versões dos produtos COTS

- Interfaces mantidas sobre controle da empresa

Uma interface que um produto COTS oferece aos desenvolvedores é chamada de Application Program Interface (API), que pode ser do tipo proprietária (se utilizar padrões próprios do fabricante) ou do tipo aberta (se utilizar padrões mundiais).

O analista do sistema deve ter controle sobre as interfaces de cada produto COTS, e evitar depender das interfaces oferecidas e controladas pelo fabricante do produto COTS. Isso porque controlando as interfaces entre os produtos, o analista pode trocar versões de um ou mais produtos COTS, garantindo que as interfaces são as mesmas, minimizando assim o impacto das trocas e reconfiguração.

Esse controle pode ser feito através de padrões de arquitetura como: wrappers , façades ou mediator, definidos no capítulo 2.

- Gerenciar as dependências entre os produtos

Todas dependências entre produtos devem ser identificadas, e estratégias devem ser analisadas para gerenciar essas dependências, evitando que uma alteração feita em um produto COTS, tenha impacto negativo sobre outros produtos do sistema COTS. O autor cita alguns tipos de dependências: sintática (parâmetros de rotinas), comportamental (interação entre produtos) e de recursos (utilização simultânea do mesmo recurso por vários produtos diferentes).

- Esforço mínimo de construção.

Devido às freqüentes alterações que um sistema COTS exige e a rapidez com que elas devam ser implementadas, o autor recomenda evitar processos complexos para gerar o sistema COTS, diminuindo o esforço, custo e tempo envolvido.

4.4.Atividades para manutenção de Sistemas COTS segundo Carney et al

1. Versionamento de arquivos segundo Carney et al em [CAR00]

Versões de todos os produtos usados a qualquer ponto do desenvolvimento do sistema devem ter sua versão arquivada. Inclui-se neste item todos os softwares necessários a equipe de manutenção que permitam modificar, recriar e testar o sistema, e que deve ser preservada enquanto houver uma versão operacional do sistema. São considerados alguns exemplos de itens:

- Versões dos produtos COTS utilizados no sistema
- Versões do *gluecode*
- Sistema Operacional
- Bibliotecas, DLLs, ... desenvolvidas internamente ou não
- documentação completa (especificações do sistema, requisitos, testes, troubleshootings, manuais do fornecedor)
- Dados do fornecedor (termos do contrato)
- informação da licença de cada produto, como por exemplo, o tempo em que irá expirar e qual o tipo de utilização (servidor, cliente)
- informação sobre o tipo de suporte de cada produto (qual tipo de comunicação, tempo de garantia)
- versão e conteúdo de patches de cada produto
- ferramentas de testes, compilação, design, modelagem necessárias para compilar, redefinir arquitetura caso haja necessidade.

Mas não basta simplesmente armazenar todas as versões de todos os produtos aleatoriamente. Devem ser definidos um método e uma ferramenta adequados que possibilitem :

- Testar a integridade de todos os dados antes que eles sejam armazenados
- A recuperação rápida de todos os dados armazenados
- Definir um controle rigoroso de entrada e retirada de dados, para que toda informação inserida e extraída do repositório de dados seja considerada confiável.
- Rastrear qual versão do sistema o item está relacionado. Desse modo é possível identificar sob quais versões do sistema serão afetadas em eventuais trocas ou correções dos produtos que compõe o sistema.

2. Justificar todas as modificações feitas no sistema COTS

Segundo os autores, independentemente das razões pelas quais um produto COTS é alterado no sistema, existem duas tarefas essenciais que a equipe de desenvolvimento e manutenção devem fazer: registrar a causa e o impacto de qualquer mudança feita nos produtos COTS que integram o sistema COTS.

Mas segundo os autores, esse registro deve ser feito antes da realização da mudança propriamente dita, e devem ser documentadas informações como: a causa que gerou a alteração, as razões pelas quais nenhuma outra abordagem é possível, uma descrição precisa da modificação planejada, o impacto sobre o sistema, as ferramentas necessárias, a infra-estrutura alterada, treinamentos necessários e evidências de que o fabricante foi consultado sobre os efeitos das modificações (caso necessário).

Após a alteração ter sido realizada, é necessário documentar: a descrição técnica da alteração, os resultados dos testes feitos (incluindo a verificação de que o produto modificado passou satisfatoriamente nos testes), as versões utilizadas de todas as ferramentas utilizadas para fazer a modificação, identificação das pessoas da organização que realmente fizeram a alteração.

3. Definir explicitamente a responsabilidade da manutenção para cada produto COTS

Devem ser definidas explicitamente quais tarefas serão de responsabilidade do fornecedor, como por exemplo: fazer reparos, relatar erros (área de suporte), enviar novas versões dos produtos COTS e enviar previsão de custo destas atividades. Um bom acordo com o fabricante deve ser feito para assegurar recebimento de suporte deve assegurar informações como:

- O tempo total da garantia
- Qual o tipo de suporte
- Prazo máximo para solução dos problemas
- Freqüência de novas versões
- Custo de novas versões

Além disso, os autores também recomendam a avaliação dos riscos ao instalar novas versões de produtos COTS no sistema.

4.5. Métricas para Sistemas COTS

Sedigh-Alin et al em [SED01] definem algumas métricas para Sistemas COTS divididas em três categorias : gerenciamento, requisitos e qualidades.

1. Gerenciamento

- Custo => custo total do sistema, incluindo custo de seleção, avaliação, integração dos produtos, licenças, recursos humanos e ferramentas.
- Tempo de envio ao mercado (time to market) => Tempo gasto entre o início do desenvolvimento e aquisição dos produtos COTS até a entrega final
- Ambiente de engenharia de software => Capacidade e maturidade do ambiente de desenvolvimento, ou seja, mede a capacidade que a equipe de desenvolvimento tem de produzir sistemas com alta qualidade. Este item baseia-se no SA-CMM (Software Acquisition Capability Maturity Model) [CHR03].
- Utilização de recursos do sistema => Percentagem de utilização de cada recurso em relação a capacidade total.

2. Requisitos

- Conformidade com os requisitos => Aderência do sistema COTS aos requisitos
- Estabilidade dos requisitos => Mudanças nos requisitos do sistema

3. Qualidade

- Adaptabilidade => Habilidade do sistema em adaptar-se as mudanças dos requisitos
- Complexidade => Complexidade das interfaces dos produtos e do código de integração (gluecode)
- Cobertura dos testes => Fração do sistema consideradas satisfatória e/ou que foram totalmente testadas
- Quantidade de falhas => Número cumulativo das falhas detectadas
- Confiabilidade => Probabilidade de operar sem falhas em períodos de tempo especificados
- Satisfação do cliente => Grau de conformidade do sistema com as expectativas do cliente.

Capítulo 5

Resultados

Como visto no capítulo anterior, existe uma série de abordagens diferentes sobre os vários aspectos relacionados com a manutenção de sistemas COTS. Fazendo uma análise delas, observamos que Clapp e Taub [CLA98] apresentam várias atividades voltadas para o processo de manutenção, enquanto que os outros trabalhos citados tratam de aspectos técnicos referentes ao processo de desenvolvimento.

Neste capítulo, organizamos estas abordagens e selecionamos aquelas que pudessem minimizar os riscos citados no capítulo 3. Elas são apresentadas aqui como "estratégias" e estão agrupadas em seis níveis de abstração: Requisitos, Planejamento, Arquitetura, Testes e Manutenção, Gerência de configuração e Controle de Qualidade. Estes níveis foram selecionados por estarem vinculados a fase do desenvolvimento em que cada estratégia deve ser implementada, uma vez que acreditamos que a preocupação com a viabilidade da manutenção do sistema COTS deve ser ocorrer durante todo o processo de desenvolvimento e não somente quando o processo de manutenção é iniciado.

Em seguida associamos cada uma das estratégias aos riscos minimizados por ela. Mas além de apresentar essas estratégias, procuramos neste capítulo entender, de modo mais objetivo, como a manutenabilidade do sistema poderia ser afetada, e adicionamos a essa associação a definição de Mari et al [MAR03] sobre as características da manutenabilidade.

Apresentamos no final deste capítulo essa associação entre cada estratégia, seus riscos minimizados e o modo pelo qual a manutenabilidade é afetada. O objetivo desta associação é permitir que a equipe do projeto possa se previnir dos riscos envolvidos, decidindo quando e como melhorar as várias faces da manutenabilidade, contribuindo desse modo para um sistema COTS de baixo custo derivado de uma fase de manutenção adequada.

5.1. Requisitos

O processo de análise de requisitos para desenvolvimento de sistemas em COTS possui algumas diferenças em relação ao processo tradicional, pois a tendência é gastar menos tempo codificando e mais tempo selecionando e avaliando produtos COTS. Apesar de ser claramente uma atividade de Engenharia de Requisitos, a avaliação correta de um produto antes que ele seja incorporado ao sistema, atinge diretamente a sua manutenção, uma vez que a capacidade de incluir ou excluir funcionalidades, monitorar falhas de um sistema, depende diretamente das capacidades de cada produto COTS.

1. Documentação do método para identificar quais requisitos serão implementados por produtos COTS

Uma das recomendações do CMMI [CHR03] é que sejam identificados na fase de levantamento de requisitos, quais requisitos serão satisfeitos por produtos COTS dentro da(s) solução(ções) arquitetural(is) proposta para o sistema COTS.

Outra recomendação do modelo é que os critérios utilizados para essa decisão devem seguir um método de tomada de decisão, definindo e registrando os motivos pelos quais determinados requisitos serão atendidos através dos produtos COTS ao invés de serem desenvolvidos internamente. Segundo [CHR03], essa documentação é necessária para as alterações do produto, independente em que fase o projeto esteja.

2. Documentação do método para seleção de produtos

Uma vez identificados quais requisitos serão implementados por produtos COTS, deve-se evitar que quaisquer produtos COTS sejam escolhidos a esmo, e para tanto é necessário utilizar um método para selecionar os produtos COTS. Esse método deve permitir uma análise de requisitos profunda que possibilite a equipe de manutenção fazer trocas de produtos similares com a utilização das mesmas técnicas empregadas, além de possibilitar a equipe de desenvolvimento avaliar a real necessidade da utilização de produtos COTS, tendo em vista suas vantagens, desvantagens e riscos.

Reunimos neste trabalho as variadas considerações descritas no capítulo 5 referentes à seleção de produtos COTS, adicionamos a experiência da autora e sugerimos então um método para seleção de produtos COTS.

Um bom processo de seleção de produtos COTS deve avaliar os seguintes itens:

- Requisitos do sistema. Identificar quais são os requisitos do sistema que serão cobertos pelo(s) produto(s) COTS. [CHR03]
- Restrições do sistema. Identificar quais são as restrições do sistema que possam definir critérios para a seleção dos produtos (certificações exigidas pela empresa, certificações exigidas pelo cliente, cont.ratos de parceria, licitações do governo, editais). [CHR03]
- Política da organização: Identificar se alguma das políticas da organização exige a utilização de parcerias pré-estabelecidas, ou restringe o uso de produtos de determinados fabricantes (concorrentes).
- Reputação do fabricante referente à qualidade de seus produtos (tempo de mercado, outros produtos que o mesmo fabricante disponibiliza). É recomendável que a empresa mantenha uma lista de fabricantes preferenciais e que estejam associados a uma avaliação dos mesmos. A escolha de produtos COTS pode ser iniciada a partir desta lista.
- Estimativa de preço: Deve ser identificada através de pesquisa de mercado ou outro método, uma estimativa dos preços dos produtos COTS existentes no mercado. Esta estimativa deve levar em conta também o gasto que a empresa está disposta a investir no projeto. Essa análise de custo possibilita avaliar se o custo enviado pelos vários fabricantes está no orçamento previsto, ou se o custo de um determinado produto está muito acima ou muito abaixo da média do mercado e deve ter, portanto, sua compra inviabilizada [CHR03].
- Disponibilidade de licenças-teste. Verificar se o fornecedor do produto disponibiliza licenças para teste, durante quanto tempo e seus custos associados.
- Qualidade da documentação: se ela está disponível, consistente, atualizada, descreve os erros já conhecidos do produto.
- Política de suporte ao usuário do produto COTS: se existe uma central de atendimento, qual o tempo médio de correção de erros, se técnicos são disponibilizados para detecção de falhas, como é o canal de comunicação, se as eventuais correções dos produtos são enviadas gratuitamente ou mediante um determinado custo.
- Política de suporte a versões antigas: verificar se existe política de lançamento de novas versões e como é o suporte a versões antigas.
- Facilidade de aprendizado. A usabilidade da ferramenta e a qualidade dos manuais podem servir como referencias neste item, que geralmente são avaliados por analistas mais experientes da empresa.

- Os custos, prazos e tipo de treinamentos necessários para manipular a ferramenta. É importante verificar se existem treinamentos específicos para determinadas ações como por exemplo, administrar e utilizar a ferramenta e seus custos associados.
- Custo e tipo de hardware necessário para utilizar o produto. Verificar se os recursos físicos são compatíveis com os produtos COTS ou devem ser trocados.
- Tipo de interfaces fornecidas : Verificar se as interfaces fornecidas seguem padrões públicos ou se são bibliotecas proprietárias (APIs), e podem ser codificadas em linguagens de programação já conhecida equipe técnica.
- Capacidade de customização do produto: É a habilidade do produto ser alterado para satisfazer um requisito ou um conjunto deles. Caso o fornecedor permita alterações, deve ser avaliado o tipo de linguagem, se é padronizado ou não.
- Funcionalidades exportáveis : identificar quais as funcionalidades do produto COTS podem ser programadas através da sua interface. Isso define como o produto será integrado no sistema COTS, possibilitando o controle dos dados de entrada e saída de cada produto COTS.
- Segurança: quais tipos e em que nível é feita a segurança do produto COTS. Interfaces com o usuário, acesso ao banco de dados, bloqueio e liberação de acesso aos arquivos e APIs devem ter seu nível de segurança verificadas.
- Custo de novas licenças : verificar o custo de licenças de desenvolvimento e produção (entregue ao cliente). além disso deve ser verificada a periodicidade de novos lançamentos e o custo das atualizações do produto.
- Compatibilidade com outros produtos : Verificar quais os outros produtos COTS do mesmo fornecedor que são compatíveis com o produto COTS em questão.
- Visibilidade interna : habilidade do produto COTS de permitir o acesso ao seu comportamento interno durante tempo de execução. Essa funcionalidade vai ser de vital importância na detecção de falhas.
- facilidade de integração com outros produtos COTS. Verificar se as versões entre os produtos COTS podem interagir entre si, lembrando que quanto maior a quantidade de produtos COTS no sistema, maior a probabilidade de surgirem incompatibilidades entre eles.
- Portabilidade: quais tipos de plataforma/sistemas operacionais que suportam o produto atualmente. Existem alguns fornecedores que garantem a portabilidade dos produtos COTS as diferentes versões do sistema operacional fornecido por ele durante um determinado tempo.

Sugerimos depois deste levantamento, a utilização de um método de tomada de decisão como o Weight Score Method descrito em [ALV01], onde são definidos pesos e prioridades

para cada um dos critérios. Cada produto é avaliado recebe uma avaliação em relação a sua conformidade com os critérios e o produto selecionado é aquele que obtiver a maior nota.

Outro método de tomada de decisão é o AHP, Analytical Hierarchy Process - [SAA90] – que decompõe um problema numa hierarquia de critérios, como mostra a figura abaixo.

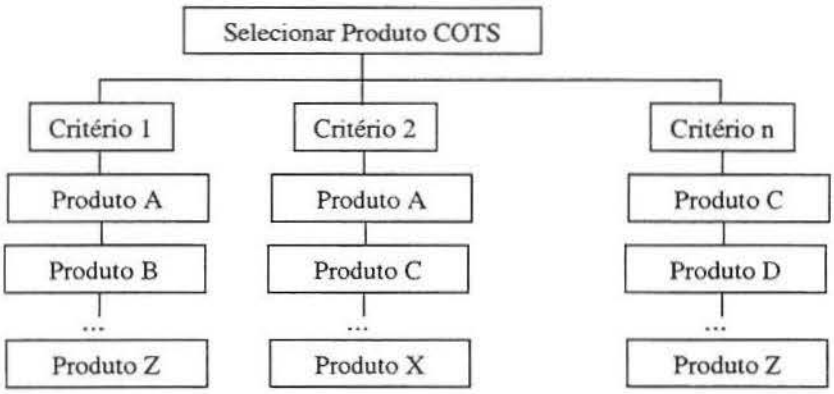

Figura 9 – Método AHP

Em seguida é determinada a importância relativa de cada critério, onde são utilizadas comparações para avaliar a importância de um critério em relação ao outro, e para tanto são utilizados pesos que se referem a intensidade da importância. Após uma série de comparações e cálculos é possível identificar quais os produtos candidatos que satisfazem melhor os critérios.

Segundo Clapp e Taub [CLA98], a tarefa de procurar produtos COTS não deve ser restrita à fase de identificação de produtos, haja vista a alta taxa em que produtos entram e saem do mercado. É necessário procurar e avaliar continuamente novos hardware e softwares disponíveis no mercado, e que poderão substituir com mais eficiência os que já existem. Muitas vezes, o desempenho pode ser melhorado com um novo hardware, ou um novo produto pode prover o sistema atual com mais funcionalidades e menos incompatibilidades. Portanto é recomendável que periodicamente o mercado seja novamente revisto, e outros produtos COTS reavaliados.

3. Documentação do método de avaliação de produtos

Independentemente do método utilizado para seleção e avaliação dos produtos COTS, deve-se registrar o resultado do método. Este registro será utilizado para:

- Documentar os motivos pelos quais os produtos COTS foram selecionados para o sistema.

- Identificar desvantagens dos produtos COTS que devem ser acompanhadas como riscos de projeto.
- Rastrear requisitos: mapear quais os requisitos do sistema são satisfeitos por quais produtos COTS.
- Analisar substituições: Caso seja necessária a substituição de um produto COTS, ele deve ter resultados similares com o produto COTS a ser substituído.
- Melhoria do método de seleção de produtos COTS. À medida que a lista de itens é utilizada, a empresa pode identificar a necessidade de novos itens a serem avaliados, melhorando deste modo o método de seleção.
- Acionar o suporte ao produto COTS.

5.2.Planejamento

1. Planejamento de Projeto

É importante que todo sistema COTS tenha um Plano de projeto. Este plano tem por finalidade gerenciar e controlar a execução do projeto, e deve ser revisado e aprovado pelos papéis competentes [CHR03]. Em relação a sistemas COTS, um plano de projeto deve prever:

- Periodicidade da troca do COTS:

  Caso o fabricante tenha uma periodicidade fixa, ou houve um acordo com a empresa, de emissão de novas versões dos produtos COTS o Plano de Projeto deve registrar o momento previsto para essas novas emissões e seu impacto no projeto.
- Entrega dos produtos COTS

  Definir e acompanhar no cronograma de projeto quando serão entregues os produtos COTS, definindo datas para avaliação da entrega. Pode ser que algumas entregas de produtos estejam vinculadas a algum tipo de pagamento, e devem estar disponíveis os valores acordados.
- Definição de critérios de aceitação dos produtos COTS

  Definir critérios objetivos para que o produto COTS seja aceito no projeto. Neste item devem ser planejados quais critérios, quando e como eles serão aplicados.
- Treinamentos necessários

  Identificar quais os treinamentos necessários para todos os membros da equipe, inclusive para manipulação de ferramentas. Além disso devem estar planejados quando serão oferecidos e quais os fornecedores destes treinamentos.

- Estratégia de comunicação com o fabricante.

  Definir um acordo formal, um contrato, com o fabricante do produto estabelecendo custo das licenças, tipo de suporte e custo das novas versões. Este acordo pode ser utilizado quando houver uma falta de compatibilidade entre os produtos COTS e hardware ou houver problemas na integração do produto COTS no sistema, tanto na fase de desenvolvimento quanto na fase de manutenção.

2. Análise de riscos

Rose cita em [ROS03], a importância de manter um plano de riscos desde o início da fase de desenvolvimento até a fase de manutenção do produto, onde os riscos relacionados aos produtos COTS são identificados, estipulados sua probabilidade de ocorrência, definidas ações preventivas (para que não ocorram) e corretivas (caso o risco se concretize).

Uma boa prática em uma empresa, é definir um conjunto mínimo de riscos que devem ser acompanhados em todos os projetos, e que contenha itens como por exemplo:

- Desvantagens do produto COTS: Uma vez identificadas às desvantagens de cada produto, é necessário acompanhar se estas desvantagens estão ocorrendo e definir um plano de ação caso necessário.
- Necessidade de instalações de novas versões dos produtos já existentes no sistema. Neste caso Clap e Taub [CLA98] sugerem que seja feita uma análise para determinar o custo e o esforço necessário (maquinário, banco de dados, interfaces, plataforma), lembrando que não realizar as alterações que o fabricante lança pode significar em perder o suporte do produto e caso essa alteração se faça necessário no futuro, envolverá o esforço de ajustar o sistema a não somente uma, mas a várias versões podendo exigir para tanto, um custo mais elevado do que fazer a troca para cada versão.
- Possibilidade do fornecedor não estar disponível no mercado.
- Licenças e máquinas devem estar disponíveis para as respectivas equipes. Há o risco da equipe estar disponível mas as licenças e maquinas não (ou vice-versa).
- Quais maquinários serão afetados : disco, memória, conexão, ambiente de testes.
- Qual impacto sobre o desempenho e a confiabilidade do sistema COTS.

O autor cita ainda, a necessidade de reavaliar esses riscos periodicamente ao longo do ciclo de vida do sistema devido às várias alterações que os produtos possam sofrer durante todo esse processo, haja vista que essas alterações podem variar desde pequenos releases até alteração de toda plataforma.

Existem várias estratégias para o acompanhamento destes riscos, como por exemplo a gerência de Risco – nível 3 em [CHR03].

2. Planejamento dos custos

A empresa deve gerenciar todos os aspectos financeiros de adquirir, usar, receber evoluções de um produto COTS, afinal de contas, uma das grandes vantagens de utilizar COTS é a redução de custos relacionados ao desenvolvimento.

Desse modo se não houver um planejamento detalhando todos os custos referentes aos produtos COTS, a organização pode gastar mais tempo e dinheiro resolvendo incompatibilidades, corrigindo falhas e pagando licenças que seria preferível desenvolver o código internamente.

Devem ser avaliados os impactos nos custos do projeto dos seguintes itens :

- Licenças : quem deverá comprá-las, quem irá mantê-las, quais versões serão compradas, custo das renovações, periodicidade de evoluções do software, qual tipo de hardware associado, tempo de uso, número de cópias, tipo do serviço de suporte [VIG97].
- Recursos humanos : necessidade de treinamentos específicos para manipular os COTS, pessoas para efetuar pesquisa contínua do mercado, análise de risco[VIG97].
- Hardware : Freqüentemente novos hardwares são lançados no mercado e muitas vezes a organização deve acompanhar estas mudanças [VIG97].
- Ferramentas : ferramentas para gerenciar a configuração do sistema, realizar testes [VIG97].
- Treinamentos : treinamentos necessários para manipular configurar, alterar, codificar a interface dos produtos COTS do sistema.

Para um planejamento mais detalhado sobre custos, existem alguns métodos de estimativas como o COCOTS [ABT00] que fornece uma abordagem detalhada para estimar os custos envolvidos no desenvolvimento de sistema COTS. Ele é subdividido em quatro sub-modelos para tratar as principais fontes de custo de um sistema COTS: seleção, adaptação, integração e manutenção. Para cada um destes sub-modelos, o modelo descreve atributos que devem ser analisados e suas respectivas escalas de importância. O processo de avaliação de custos é feito usando métricas para estimar o esforço necessário para realizar cada etapa do desenvolvimento do sistema COTS.

5.3.Arquitetura

1. Aumentar controle sobre as interfaces de cada produto

A interface de um produto é o modo pelo qual o produto COTS interage com o sistema e com outros produtos. Em seu trabalho, Vigder [VIG98] detalha a necessidade de manter controle sobre elas, conhecendo os detalhes do que cada produto espera de entrada e de saída.

Quando a equipe de manutenção detém controle sobre todos os dados que entram e saem de cada interface, permite que erros sejam identificados e corrigidos de maneira rápida além de minimizar o impacto no restante do sistema. Além disso ao fazer adaptações no sistema, a equipe de manutenção pode rastrear quais outros produtos serão afetados. O autor cita algumas técnicas como wrappers, façades e mediators descritos no capítulo 2.

Porém vários outros benefícios podem ser alcançados com essas técnicas como por exemplo: podem servir para adicionar ou remover funcionalidades de produtos, criar padrões proprietários de comunicação entre produtos (ou convertê-los para padrões abertos), mas consideramos que sua maior contribuição para a fase de manutenção consiste em facilitar a localização de falhas do sistema COTS, pois permite que um erro seja rastreado ate sua origem que pode estar em um produto COTS, na integração entre alguns produtos COTS ou no gluecode.

Guerra et al em [GUE03] consideram o problema de integrar produtos COTS em sistemas com grande dependência dos requisitos utilizando a técnica de wrappers, para encapsular um produto adicionando a eles capacidades de tolerar falhas.

5.4.Testes e Manutenção

1. Efetuar testes de integração

Por se tratarem de verdadeiras caixas-pretas, a dificuldade de integrar produtos COTS é grande, porém existem várias formas de diminuir os riscos associados a integração de COTS, entre elas:

- A integração pode ser feita por pessoas especializadas, como por exemplo, um integrador do sistema com larga experiência em analisar o comportamento de cada produto COTS e seus inter-relacionamentos.
- Obter suporte do fabricante durante a integração caso hajam erros constatadamente relacionados ao produto, observando que esse item deve estar coberto no contrato com o fabricante.
- Iniciar os testes de integração desde o início do desenvolvimento [LAW01]

- utilizar o método CISD [HIS98] que propõe a utilização de protótipos para analisar esse tipo de problemas ainda na fase de seleção e avaliação de produtos.

2. Retestar o sistema ao instalar novas versões de produtos COTS

Ao inserir novas versões de produtos COTS no sistema, é ideal que ele seja totalmente retestado para avaliar se os requisitos do sistema COTS continuam sendo mantidos e o desempenho do novo produto. Técnicas como black box citada por Vigder [VIG98] podem ser utilizadas para verificar o impacto das alterações em todas as funcionalidades do sistema.

3. Monitorar o sistema

Vigder em [VIG97] sugere monitorar o comportamento do sistema para analisar desempenho e possíveis anomalias no sistema COTS. Essa análise pode ser feita de várias maneiras: utilizando o controle sobre as interfaces (caso tenham sido utilizados padrões de projeto durante o desenvolvimento do sistema, ou caso o fabricante do produto COTS forneça acesso às interfaces do produto COTS), utilizando ferramentas adequadas que permitem que a equipe de manutenção detectar falhas em protocolos de comunicação, monitoração de desempenho e utilização dos recursos de hardware.

4. Definir contratos com suporte dos fabricantes dos produtos

Definir contratos de suporte de cada produto COTS é uma atividade intrínseca a manutenção de sistemas COTS. Como a qualidade de cada produto bem como, a correção de erros depende somente do fabricante, é necessário que a organização estipule regras claras que definam direitos e deveres do suporte de cada produto. Além disso, é necessário clarificar como será o suporte caso : o produto COTS seja descontinuado, o fabricante do produto COTS falir e não haja instalações de novas versões.

5.5.Gerência de Configuração

1. Definir as atividades de Gerência de Configuração e suas responsabilidades

Durante a fase de desenvolvimento do sistema COTS devem ser definidas as seguintes atividades devem ser feitas para cada versão do sistema COTS:

- Selecionar quais os produtos intermediários e finais do sistema serão versionados e devem ser alterados somente sob autorização (documentação, código-fonte, códigos-binários). É importante ressaltar que essa autorização é necessária para evitar que alterações sejam feitas a esmo, sem nenhum tipo de análise de impacto em custo e tamanho do sistema, ou mesmo que elas não estejam associadas ao interesse da empresa [CHR03].
- Definir explicitamente a responsabilidade pelos procedimentos de gerência da Configuração [CHR03].
- Definir explicitamente a responsabilidade pela criação de novas versões/releases do sistema COTS [CHR03].
- Descrever a versão, o nome, e o ambiente de todas as ferramentas utilizadas no desenvolvimento. É importante que sejam descritas as versões de compiladores, bibliotecas, bancos de dados, ferramentas de integração de produtos COTS e as ferramentas de gerência para que a equipe de manutenção possa recuperar totalmente o ambiente em que o sistema COTS foi desenvolvido [CHR03][CAR00].
- Definir um repositório de dados padrão para armazenar todos os produtos gerados pelo projeto.
- Definir quais os níveis de acesso permitidos a este repositório, ou seja, as inclusões/remoções/atualizações dos dados deste repositório podem ser efetuadas somente após autorizações dos responsáveis . Isto é necessário para garantir que o repositório esteja consistente, garantindo que todos os produtos ali contidos estão corretos [CHR03].
- Manter um mapeamento entre todas as autorizações de mudanças e os dados que foram alterados para que todas as informações contidas no repositório possam ser rastreadas [CHR03].
- Definir a nomenclatura padrão da empresa, evitando que nomes aleatórios sejam dados aos documentos do projeto.
- Listar em quais clientes foram instaladas as versões do sistema COTS.
- Documentar qual versão de cada produto COTS que compõe o sistema COTS, como eles implementam os requisitos do sistema COTS e qual o relacionamento de dependências entre eles [CHR03].
- Documentar a versão do *gluecode* que compõe o sistema COTS [CAR00]
- Listar os erros conhecidos e resolvidos.
- Listar os erros conhecidos e não resolvidos.
- Manter a documentação técnica atualizada (lista de requisitos do sistema COTS, solução de arquitetura, especificações técnicas, listagem dos testes, lista e data de vencimento das licenças dos produtos COTS e o suporte de cada produto COTS) e armazenadas no repositório do projeto [CAR00].

Uma vez recuperadas corretamente as informações detalhadas do projeto, as versões corretas de todas as ferramentas utilizadas e as versões dos produtos COTS, a equipe de manutenção pode avaliar com mais rapidez e eficácia as alterações exigidas e decidir por efetuar ou não correção.

2. Documentar todas as alterações a serem feitas nos produtos COTS

Carney et al em [CAR00] afirmam que justificar de modo formal todas as alterações antes e depois que elas sejam feitas contribui de várias maneiras para a manutenção.

A primeira delas é que tanto a equipe de manutenção quanto a de desenvolvimento ao justificar antes as modificações, fazem o planejamento e análise de riscos e impacto, levando em conta critérios exigidos pela organização, evitando idéias individuais e alterações desnecessárias.

Outro grande benefício é que a documentação dos motivos, das restrições, das alterações de arquitetura, permite que a equipe de manutenção compreenda as necessidades das alterações feitas.

Os autores ainda sugerem documentar antecipadamente informações como: a causa que gerou a alteração, uma descrição precisa modificação planejada, evidências de que o fabricante foi consultado sobre os efeitos das modificações caso necessário.

E após a alteração ter sido realizada, deve ser documentado : a descrição técnica da alteração, todos os dados dos testes feitos, incluindo verificação de que o produto modificado passou satisfatoriamente nos testes, versões utilizadas de todas as ferramentas utilizadas para fazer a modificação e identificação das pessoas da organização que realmente fizeram a alteração.

5.6. Controle de Qualidade

1. Manter banco de dados de métricas

Algumas métricas citadas na seção 4.4 podem contribuir para a qualidade do sistema.

[CHR03] cita que todas as métricas identificadas para a organização devem estar vinculadas aos objetivos de cada uma bem como aos métodos de coleta e análise.

2. Manter acordo com fabricantes dos produtos COTS

Clap e Taub em [CLA98] recomendam manter um acordo com o fabricante do produto, geralmente através de licenças, para determinar a correção dos eventuais problemas.

Os autores ressaltam que uma das grandes vantagens dos produtos COTS é a grande quantidade de usuários ao redor do mundo que ajudam a encontrar problemas no produto de modo que a qualidade do mesmo tende a aumentar. É muito comum o fabricante do produto COTS, periodicamente agrupar a resolução destes problemas e enviar aos usuários do produto uma versão contendo as correções. Nestes casos, a equipe de manutenção do sistema COTS deve estar apta a decidir se essa nova versão deve ou não ser instalado, tendo em vista que, os erros corrigidos nela podem afetar gravemente o sistema, e definir quais as partes de sistema devem ser retestadas para manter a qualidade atual do sistema COTS.

Além disso, outro item importante associado a qualidade, é o modo pelo qual os problemas encontrados no sistema COTS são identificados, armazenados e analisados. Deve ser possível identificar quando um problema foi originado por um produto COTS e quanto tempo levará sua correção. O tipo de informação dado pelo fornecedor define a confiabilidade de cada produto, e pode ser útil quando for necessário determinar se um produto COTS deve ser trocado por suas funcionalidades ou por seu fornecedor.

5.7. Tabela de associação entre atividades x riscos x características

Nesta seção associamos para cada estratégia selecionada, os riscos minimizados e as características da manutenabilidade afetadas. Além disso mantivemos o mesmo agrupamento nos níveis de abstração tornando possível associar a fase do ciclo de vida em ocorre tanto a estratégia quanto o risco envolvido.

Essa tabela pode ser utilizada de várias formas:

- Dada uma estratégia é possível identificar quais riscos elas minimiza, identificando claramente qual o impacto da estratégia no projeto.
- Dada uma estratégia é possível identificar quais as características da manutenabilidade estão envolvidas, permitindo uma visão objetiva para a equipe de projeto de como a manutenabilidade pode ser melhorada.
- Dada uma estratégia é possível ter controle de quando ela deve ser implementada pois cada uma delas está associada a um nível de abstração referente ao momento em que ela deve ser iniciada.
- Dada uma característica da manutenabilidade, é possível selecionar quais atividades podem ser feitas para aumentá-la.
- Dado num determinado risco, é possível identificar quais as estratégias que podem ser feitas para minimizá-lo e o momento em que eles são devem ser verificados.
- Definir quais as estratégias que poderão ser realizadas mediante o custo-benefício de cada uma para garantir uma fase de manutenção adequada para o projeto, pois uma determinada estratégia pode minimizar vários riscos simultaneamente.

Podemos, por exemplo, verificar que a atividade "documentar o método para identificação de produtos", diminui vários riscos, e aumentam a flexilibilidade, reusabilidade, modificabilidade, portabilidade, extensibilidade e testabilidade.

Por outro lado, ao introduzir a estratégia de "documentar o método para identificação de produtos" estamos na verdade melhorando o nível de requisitos do sistema COTS.

Mas se for desejável que o sistema tenha sua flexibilidade aumentada, basta verificar quais atividades estão associadas a essa característica e selecionar quais delas serão implementadas.

Atividades	Riscos	Característica Da Manutenabilidade
Requisitos		
Documentação dos requisitos do sistema a serem cobertos pelos produtos COTS.	3.1.1 - Falta de técnicas específicas para seleção de produtos COTS. 3.1.2 - Falta de documentação e acompanhamento dos riscos associados à seleção de produtos COTS. 3.1.4 - Versões diferentes de COTS oferecem maior/menor número de funcionalidades. 3.1.6 - Níveis de segurança diferentes entre produtos COTS 3.1.7 - Rastreabilidade entre requisitos do sistema e produtos COTS que implementam estes requisitos. 3.1.8 - Versões de COTS com uso restrito 3.2.2 - Incompatibilidade entre produtos COTS do mesmo sistema. 3.3.1 - Poucos produtos COTS oferecem ferramentas para rastrear e monitorar erros 3.3.8 - Documentação escassa do produto COTS	Flexibilidade Reusabilidade Modificabilidade Portabilidade Extensibilidade

	3.5.1 - Falta de padronização da qualidade do produto COTS. 3.5.2 - A correção de falhas encontradas depende do interesse do fabricante do produto COTS	
Documentação do método para seleção de produtos COTS	3.1.1 - Falta de técnicas específicas para seleção 3.1.4 - Versões diferentes de COTS oferecem maior/menor número de funcionalidades. 3.1.6 - Níveis de segurança diferentes entre produtos COTS 3.1.5 - Vírus contidos em produtos COTS 3.1.6 - Níveis de segurança diferentes entre produtos COTS 3.3.4 - Pouca avaliação sobre o impacto das alterações no sistema COTS no ambiente do usuário. 3.3.5 - Falta de planejamento ao integrar componentes 3.3.6 - Incompatibilidade entre versões do produto COTS com o sistema. 3.3.7 - Utilização de novos padrões pode gerar retrabalho. 3.3.8 - Falta de documentação atualizada.	Flexibilidade Reusabilidade Modificabilidade Portabilidade Extensibilidade Testabilidade
Documentação do método de avaliação	3.1.1 - Falta de técnicas específicas para seleção de produtos COTS.	Flexibilidade

	3.1.4 - Versões diferentes de COTS oferecem maior/menor número de funcionalidades. 3.3.6 - Incompatibilidade entre versões do produto COTS com o sistema. 3.3.7 - Utilização de novos padrões pode gerar retrabalho.	Reusabilidade Modificabilidade Extensibilidade Portabilidade Testabilidade
Planejamento		
Planejamento de Projeto	3.2.1- Periodicidade da troca do COTS 3.2.2 – Incompatibilidade entre produtos COTS do mesmo sistema. 3.2.4 – há falta de treinamento para manipular os produtos COTS 3.2.6 - Estratégia de comunicação com o fabricante 3.3.3 – Alterações são feitas no código sem justificativas prévias. 3.3.5 -Problemas na integração do produto COTS no sistema	Flexibilidade Reusabilidade Modificabilidade Portabilidade Extensibilidade Testabilidade
Análise de riscos	3.1.2 – Falta de documentação e acompanhamento dos riscos associados à seleção de produtos COTS. 3.2.3 – Falta de planejamento dos custos 3.2.5 - Gerência para análise de riscos	Modificabilidade Portabilidade Extensibilidade Testabilidade Integrabilidade

Planejamento dos custos	3.2.1 - Periodicidade da troca do COTS 3.2.4 - Há falta de treinamento para manipular os produtos COTS 3.2.3 - Falta de planejamento dos custos 3.3.7 - Utilização de novos padrões pode gerar retrabalho.	Modificabilidade Integrabilidade
Arquitetura		
Aumentar controle sobre as interfaces de cada produto	3.1.5 - Vírus contidos em produtos COTS. 3.3.1 - Poucas produtos oferecem rastreamento de erros. 3.3.2 - Pouca utilização de arquiteturas apropriadas para sistemas COTS 3.3.5 - Falta de planejamento ao integrar componentes 3.3.6 – Incompatibilidade entre versões dos produtos COTS e o sistema. 3.3.7 - Falta de utilização de novos padrões pode gerar retrabalho. 3.4.4 - Dificuldade em identificar e corrigir corretamente falhas.	Testabilidade Integrabilidade Rastreabilidade Monitorabilidade

Testes e Manutenção		
Efetuar testes de integração	3.1.4 - Versões diferentes de COTS oferecem maior/menor número de funcionalidades. 3.1.5 - Vírus contidos em produtos COTS 3.3.4 - Pouca avaliação sobre o impacto das alterações no sistema COTS no ambiente do usuário. 3.3.5 - Falta de planejamento ao integrar componentes. 3.3.6 - Incompatibilidade entre versões dos produtos COTS e o sistema. 3.4.3 - Falta de consistência nos testes	Testabilidade Integrabilidade Monitorabilidade
Retestar o sistema ao instalar novas versões de produtos COTS.	3.1.4 - Versões diferentes de COTS oferecem maior/menor número de funcionalidades. 3.3.5 - Falta de planejamento ao integrar componentes	Testabilidade Integrabilidade
Monitorar o sistema.	3.4.4 - Dificuldade em identificar e corrigir corretamente falhas	Monitorabilidade
Definir contratos com suporte dos fabricantes dos produtos	3.1.2 - Falta de documentação e acompanhamento dos riscos associados à seleção de produtos COTS. 3.1.3 - Os requisitos dos produtos COTS está sob controle do seu fabricante.	Flexibilidade Modificabilidade Portabilidade Extensibilidade

	3.1.4 - Versões diferentes de COTS oferecem maior/menor número de funcionalidades. 3.2.2 - Incompatibilidade entre produtos COTS do mesmo sistema. 3.2.6 - Estratégia de comunicação com o fabricante. 3.4.1 - Estrutura do suporte do fabricante 3.4.2 - Falta de gerência do contrato de manutenção. 3.4.3 - Falta de consistência nos testes 3.5.1 - Falta de padronização da qualidade dos produtos COTS de um mesmo sistema COTS. 3.5.2 - A correção de falhas encontradas depende do interesse do fabricante do produto COTS	Testabilidade
Gerência de Configuração		
Definir as atividades de Gerência de Configuração e suas responsabilidades.	3.1.7 - Rastreabilidade entre requisitos do sistema e produtos COTS que implementam estes requisitos. 3.6.1 - As diferentes versões dos produtos não tem uma identificação clara de seu conteúdo. 3.6.2 - Ferramentas de desenvolvimento e manutenção diferentes	Integrabilidade
Documentar todas as alterações a serem	3.1.7 – Rastreabilidade entre requisitos do	Integrabilidade

feitas nos produtos COTS.	sistema e produtos COTS que implementam estes requisitos. 3.3.3 - Alterações são feitas no código sem justificadas prévias. 3.4.4 – Dificuldade em identificar e corrigir corretamente falhas	
Controle de Qualidade		
Manter Banco de Dados de Métrica	3.4.4 – Dificuldade em identificar e corrigir corretamente falhas 3.5.1 – Falta de padronização da qualidade dos produtos COTS de um mesmo sistema COTS.	Monitorabilidade
Manter acordo com fornecedores	3.1.3 – Os requisitos dos produtos COTS está sob controle do seu fabricante. 3.4.1- Estrutura do suporte do fabricante 3.4.2 - Inexistência do suporte do fornecedor 3.4.3 – Falta de consistência nos testes 3.5.1 – Falta de padronização da qualidade dos produtos COTS de um mesmo sistema COTS. 3.5.2 - A correção de falhas encontradas depende do interesse do fabricante do produto COTS.	Flexibilidade Reusabilidade Modificabilidade Portabilidade Extensibilidade Testabilidade

Figura 10 - Tabela de associação entre atividades x riscos x características

Capítulo 6

Conclusões e trabalhos futuros

Produtos COTS nos seduzem com a vantagem de usufruir o que existe de mais moderno em termos de tecnologia existentes no mercado em menor tempo e custo que o desenvolvimento tradicional permitiria, pois a idéia de simplesmente agregar produtos com inúmeras funcionalidades em um único sistema, parece tornar mais simples as tarefas das equipes de desenvolvimento e de manutenção.

Mas como foi apresentado, agregar produtos em um único sistema não é uma tarefa simples. Muito pelo contrário, traz consigo uma série de riscos que devem ser gerenciados pois caso contrário podem inviabilizar a manutenção de um sistema COTS, e exigem portanto cautela em sua utilização.

Após um estudo mais detalhado sobre as possíveis estratégias para melhorar a fase de manutenção de um sistema COTS, concluímos que a preocupação com a manutenção deve estar presente em todas as fases do desenvolvimento do sistema COTS, iniciando nos processos de seleção e avaliação de produtos COTS, passando pelo desenvolvimento de arquiteturas flexíveis que facilitem a troca de produtos. Além disso é necessária uma gerência de configuração e controle de qualidade que permitam o versionamento e a qualidade dos itens do projeto respectivamente.

Nossa contribuição foi apresentar não somente algumas destas estratégias, mas também identificar quais os riscos que elas podem minimizar, definir a fase em que elas devem ser incorporadas ao projeto (6 níveis de abstração) e entender o modo que elas afetam a manutenabilidade de um sistema COTS (características da manutenabilidade). Esses procedimentos permitem que a equipe do projeto faça um planejamento mais detalhado do custo, esforço e tempo necessários para melhorar a manutenção tanto a curto quanto a longo prazo, além de compreender de modo mais objetivo o modo pelo qual a manutenabilidade está sendo aprimorada, evitando divagações sobre tantas possibilidades uma vez que melhoria da manutenção está permeada por todo o desenvolvimento do sistema.

Outra importante contribuição é permitir que a equipe do projeto tenha consciência dos riscos que envolvem a construção de sistemas COTS, possibilitando que a empresa identifique quais deles representam uma ameaça para o projeto e deverão portanto ser

tratados. Além disso, a empresa pode utilizar este trabalho para gerar uma lista de riscos comum a todos os projetos onde os riscos mais prováveis são acompanhados periodicamente nas reuniões de projeto, proporcionando um acompanhamento padrão que reflita a preocupação da empresa em relação à qualidade dos sistemas. Além de chamar a atenção para os riscos, este trabalho também fornece várias opções de estratégias para contê-los, e após a avaliação do custo de cada uma em relação aos benefícios trazidos para o projeto, a empresa pode decidir quais das estratégias serão realizadas e em que fase do ciclo sua execução está associada. Estas informações são imprescindíveis para promover a gerência de riscos adequada ao projeto beneficiando de várias formas a empresa: evita que estes riscos ocorram inesperadamente durante o projeto, permite uma estimativa de custo e esforço para a contenção destes riscos, e além disso, concentra o esforço da equipe de projeto na prevenção dos riscos e não mais na correção deles diminuindo o custo total do projeto.

Atenção especial deve ser dada aos riscos que envolvem a integração dos produtos COTS. Eles surgem como os mais prováveis e de alto impacto no projeto pois a inserção ou alteração de qualquer produto pode comprometer toda a estrutura do sistema COTS, e caso elas não sejam feitas de modo planejado e controlado podem reduzir em muito a economia gerada pela compra dos produtos COTS. Recomendamos desse modo, que estes riscos sejam permanentemente acompanhados durante as reuniões de projeto.

Observamos que foram tratados, nesta monografia algumas estratégias e riscos, que certamente exigem mais estudos, mas além disso, algumas questões que impactam diretamente na manutenção ficaram em aberto, como por exemplo: Quais tipos de testes podem ser realizados para garantir a integridade dos sistemas após mudanças feitas no sistema ? Quais outras técnicas podem garantir uma arquitetura apropriada para obter controle das interfaces ? Existe outra abstração para o termo manutenabilidade ? Quais os relacionamentos entre as estratégias levantadas ? Quais serão os resultados deste estudo aplicados em projetos reais (case-study) ?

Portanto, as atividades aqui reunidas, estão longe de ser uma descrição completa de como garantir a manutenabilidade do sistema, mas são suficientes para ilustrar a necessidade de um estudo mais completo nesta área.

Glossário

COTS	*Commercial Off-The-Shelf* são produtos de software disponíveis no mercado que podem ser comprados ou licenciados e são suportados e atualizados pelo vendedor que retêm os direitos de propriedade.
AHP	*Analytical Hierarchy Process* é uma técnica de tomada de decisão baseada na decomposição de um problema com critérios múltiplos numa hierarquia e é feito um ranking das alternativas a partir de comparações aos pares entre elas.
Componente de software	São artefatos autocontidos, claramente identificáveis, que descrevem ou realizam uma função específica e têm interfaces claras em conformidade com um dado modelo de arquitetura de software, documentação apropriada e um grau de reutilização definido.
CBS	COTS Based Software, mesmo conceito de sistema COTS
GUI	Graphical User Interface, conjunto de telas que fazem a interface com o usuário de um produto
OSS	OpenSource Software
SCM	Software Configuration Management
API	Application Program Interface
Requisito	Uma condição ou capabilidade que deve ser contemplada por um produto para satisfazer um contrato, padrão, especificação ou outros documentos estabelecidos formalmente. IEEE 610.12-1990.
Requisito funcional	Define as funções que o sistema precisa prover.
Requisito não funcional	É uma propriedade de qualidade ou restrição que o software precisa satisfazer.
Rastreabilidade de requisitos	É evidência de uma associação entre um requisito e sua fonte, sua implementação e verificação

Bibliografia

[BAS00] – V. Basili, B. Boém, **COTS-Based Systems: A Classification of Issues**, IEEE Computer Vol.33, No. 11: November 2000

[IEE90] **IEEE** Std. 610.12, "Standard Glossary of Software Engineering Terminology", **IEEE** Computer Society Press, Los Alamitos, CA, 1990.

[BRO00] - L. Brownsword, T. Oberndorf, C. Sledge, **Developing New Processes for COTS-Based Systems**, IEEE Software, vol 17, no. 4, pp 48-55, July/August 2000,

[SCH99]- N.F. Schneidewind, A.P.Nikora, **Issues and Methods for Assessing COTS Reliability, Maintainability, and Availability,** in Proceedings of the Workshop on Ensuring Successful COTS Development , 21st International Conference on Software Engineering, Los Angeles, CA, May 1999.

[VOA00] - J. N. Voas, **Disposable COTS-Based Information Systems**, in Proceedings of the Conference of Software Maintenance and Reengineering, Zurich, Switzerland. IEEE Computer Society, 29 February – 3 March, 239-241, 2000

[VIG98] - M.R. Vigder ,J.C. Dean, **Building Maintainable COTS Based Systems**, In Proceedings of the International Conference on Software Maintenance, Washington, DC. November 1998.

[CLA98] - J. A. Clapp, A.E. Taub, **A Management Guide to Software Maintenance in COTS-Based Systems**, Eletronic Systems Center, November 1998.

[OBE03] - T. Oberndorf, **COTS and Open Systems – An Overview**, available at http://www.sei.cmu.edu/str/descriptions/cots.html visited at February 2003.

[VIG96] - M. Vigder, M. Gentleman, J. Dean, **COTS Software Integration: State of the Art**, Technical Report NRC No. 39190 1996

[VIG97] - M. Vidger, J. Dean, **An Architectural Approach to Building Systems from COTS Software Components**, In Proceedings of the 1997 Center for Advanced Studies Conference (CASCON 97), Toronto, Ontario, November, 1997

[BAS01] - V. Basili, B. Boém, **COTS-Based Systems Top 10 List**, IEEE Computer 34(5), May 2001

[CAR97] - D. Carney, **Assembling Large Systems from COTS Components: Opportunities, Cautions and Complexities**, SEI Monographs on Use of Commercial Software in Government Systems, SEI, June 1997.

[WAL98] - K. Wallnau, D. Carney, B.Poolak, **How COTS software Affects the Design of COTS-Intensive Systems**, SEI Interactive, 6/98, 1998.

[MOR02] - M. Morisio, M. Torchiano, **Definition and Classification of COTS: A proposal, In: First International Conference**, ICCBSS 2002, Orlando, Florida, February 2002.

[SZY98] - C. Szyperski, **Component Software Beyond Object Oriented Programming**, Addison-Wesley, 1998.

[HIS98] - Hissam A. Scott, **Experience Report: Correcting System Failure in a COTS Information System**, In Proceeding of International Conference of Software Maintenance – ICSM'98.

[CHR98] - R. Cherinka, C. M. Overstreet, J. Ricci, **Maintaining a COTS Integrated Solution – Are Traditional Static Analysis Techniques Sufficient for this New Programming Methodology**, In Proceeding of International Conference of Software Maintenance – ICSM'98.

[LIN98] - U. Lindqvist, E. Jonsson, **A Map of Security Risks Associates with Using COTS, IEEE Computer Society**, June 1998.

[CAR00] - D. Carney, S. A. Hissan, D. Plakosh, **Complex COTS-Based Software Systems: Practical Steps for their Maintenance**, Journal of Software Maintenance: Research and Practice, 12, 6 (2000):357-376

[HIS98] - S. Hissan, D. Carney, **Isolating Faults in Complex COTS-Based Systems,** SEI Monographs, February 1998

[TRA97] - V. Tran, D. Liu, **A Procurement-centric Model for Engineering Component-based Software Systems**, Proceedings of the Fifth International Symposium on Assessment of Software Tools - SAST'97. Jun, 1997.

[MOR00] - M. Morisio, C.B.Seaman, A. Parra, V. Basili, S. Condon, and S. Kraft, **Investigating and Improving a COTS-Based Software Development Process**, presented at 22nd International Conference on Software Engineering, Limerick, Ireland, 2000

[KON96] - Kontio J., **A Case Study in Applying a Systematic COTS Selection**, Proceedings of the 18th International Conference on Software Engineering, IEEE CS Press, Marco 1996.

[VIG98] - M. Vigder, **An Architecture for COTS Based Software Systems**, NRC Report 41603, National Research Council of Canada, Institute of Information Technology, November 1998.

[MAR03] - M. Mari, N. Eila **The Impact of Maintainability on Component-based Software Systems**, In Proceeding of the 29th Euromicro Conference, 2003.

[SOM03] - I. Sommerville, **Software Engineering**, 6th edition, Addison-Wesley, 2003

[KON96] - J. Kontio, G. Caldiera, V. R. Basili, **Defining Factors, Goals and Criteria for Reusable Component Evaluation**. In Proceedings of CASCON'96, pp17-28, November 1996.

[KON96a] - J. Kontio, **A Case Study in Applying a Systematic Method for COTS Selection**, In Proceedings of the18th International Conference on Software Engineering (ICSE), IEEE Computer Society, 1996.

[LAW01] - P.K. Lawlis, K.E.Mark, D.A. Thomas, T. Courtheyn, **A Formal Process for Evaluating COTS Software Products**,IEEE Computer 34(5):58 – 63,2001.

[ROS03] - L. Rose, Risk Management of COTS Based Systems Development, **Component-Based Software Quality, Component-Based Software Quality**, Methods and Techniques Springer LNCS vol 2693, 2003.

[SED01] - Sedigh-Ali, S., Ghafoor A., Paul R.A., **Software Engineering Metrics for COTS-based systems**, IEEE Computer 34 ,44-50, 2001

[GUE03] - P.A.C. Guerra, C.M.F. Rubira, A. Romanovsky, R.Lemos, **Integrating COTS Software into Dependable Software Architetures,** in Proceedings of the 6th IEEE International Symposium on Objetct-Oriented Real-Time Distributed Computin (ISORC'03), Hokkaido, Japan, 2003.

[SAA90] - Saaty, T. **The Analytic Hierarchy Process**. New York: McGraw-Hill, 1990

[DEL99] - R. DeLine, **A Catalog of Techniques for Resolving Packaging Mismatch**, In Proc. 5th Symposium on Software Reusability (SSR'99. Los Angeles, CA. May 1999. pp 44-53.

[ALV01] - Alves C. Frota, **Seleção de Produtos de Software Utilizando um Abordagem Baseada em Engenharia de Requisitos**, 2001. 152fs. Tese (Mestrado em Ciência da Computação) , Centro de Informática, Universidade Federal de Pernambuco.

[GAM95] – E. Gamma, R. Johnson, R. Helm, H. Vilissides, **Design Patterns: Elements of Reusabele Object-Oriented Software,** Addison Wesley, 1995.

[CHR03] – M. B. Crhissis, M. Konrad, S. Shrum, CMMI – Guidelines for process Integration and Product Improvement – Assison Wesley – 2003.

[ABT00] – C. Bats, B. W. Boehm, E. B. Clark, "Cocots: A Cots Software Integration Lifecycle Cost Model – Model Overview and Preliminary Data Collection Findings", http://sunset.usc.edu/publications/TECHRPT/2000, visited on March, 2003.